

Synthesis Tool Commands

Version Q-2019.12, December 2019

SYNOPSYS®

Copyright Notice and Proprietary Information

© 2019 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <http://www.synopsys.com/company/legal/trademarks-brands.html>.

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

www.synopsys.com

Contents

acs_check_directories	32
acs_compile_design	34
acs_create_directories	36
acs_customize_directory_structure	38
acs_get_parent_partition	39
acs_get_path	41
acs_merge_design	44
acs_read_hdl	48
acs_recompile_design	52
acs_refine_design	55
acs_remove_dont_touch	57
acs_report_attribute	59
acs_report_directories	60
acs_report_user_messages	62
acs_reset_directory_structure	64
acs_set_attribute	65
acs_submit	67
acs_submit_large	69
acs_write_html	71
add_module	73
add_parameter	76
add_pg_pin_to_db	78
add_pg_pin_to_lib	81
add_port_state	84
add_power_state	86
add_pst_state	90
add_row	92
add_state_transition	96
add_to_collection	98
add_to_rp_group	101
alias	105
alib_analyze_libs	107
all_active_scenarios	108
all_clock_gates	110
all_clocks	112
all_connected	114
all_critical_cells	116
all_critical_pins	118

all_designs	120
all_dont_touch	121
all_drc_violated_nets	123
all_fanin	125
all_fanout	127
all_high_fanout	130
all_ideal_nets	132
all_inputs	134
all_isolation_cells	136
all_level_shifters	137
all_macro_cells	138
all_outputs	140
all_physical_only_cells	142
all_registers	144
all_rp_groups	147
all_rp_hierarchicals	149
all_rp_inclusions	151
all_rp_instantiations	153
all_rp_references	155
all_scenarios	157
all_self_gates	159
all_test_modes	161
all_threestate	163
all_tieoff_cells	165
all_upf_repeater_cells	167
analyze	168
analyze_datapath	174
analyze_datapath_extraction	177
analyze_dw_power	181
analyze_minpwr_library	185
analyze_mv_design	188
analyze_rtl_congestion	192
append_to_collection	195
apply_clock_gate_latency	197
apply_power_model	199
apropos	202
as_collection	204
associate_supply_set	206
balance_buffer	208
balance_registers	210
break	213
cd	215
cell_of	217

change_link	219
change_names	222
change_selection	227
characterize	230
check_bindings	235
check_block_abstraction	237
check_bsd	239
check_budget	244
check_design	247
check_error	250
check_implementations	252
check_library	254
check_license	260
check_mv_design	262
check_rp_groups	268
check_scan_def	271
check_scenarios	274
check_synlib	276
check_target_library_subset	278
check_timing	280
check_tlu_plus_files	285
check_upf	287
clean_buffer_tree	289
close_mw_lib	291
collections	292
compare_collections	296
compare_delay_calculation	298
compare_lib	301
compare_supplies	303
compile	305
compile_exploration	312
compile_partitions	314
compile_prefer_runtime	315
compile_ultra	317
compute_polygons	322
connect_logic_net	326
connect_net	328
connect_pin	330
connect_supply_net	332
continue	334
convert_from_polygon	336
convert_pg	338
convert_to_polygon	340

copy_collection	342
copy_design	344
copy_mw_lib	347
cputime	349
create_auto_path_groups	351
create_block_abstraction	355
create_bounds	358
create_bsd_patterns	362
create_buffer_tree	366
create_bus	369
create_cache	372
create_cell	374
create_clock	377
create_command_group	380
create_core_area	382
create_design	384
create_dft_netlist	386
create_die_area	388
create_generated_clock	389
create_link_block_abstraction	393
create_logic_net	396
create_logic_port	397
create_missing_constraints	398
create_multibit	401
create_mw_lib	406
create_net	408
create_net_search_pattern	410
create_net_shape	414
create_operating_conditions	419
create_pass_directories	422
create_pin_guide	424
create_placement_blockage	427
create_port	431
create_power_domain	433
create_power_state_group	437
create_power_switch	439
create_pst	443
create_qor_snapshot	445
create_qtm_clock	449
create_qtm_constraint_arc	451
create_qtm_delay_arc	455
create_qtm_drive_type	459
create_qtm_generated_clock	462

create_qtm_insertion_delay	464
create_qtm_load_type	466
create_qtm_model	468
create_qtm_path_type	470
create_qtm_port	473
create_register_bank	475
create_route_guide	477
create_rp_group	480
create_scenario	487
create_site_row	489
create_supply_net	492
create_supply_port	495
create_supply_set	497
create_terminal	500
create_test_protocol	503
create_track	505
create_user_shape	508
create_via	512
create_via_master	516
create_via_rule	520
create_voltage_area	522
create_wiring_keepouts	525
cross_probing_filter	527
current_design	529
current_design_name	531
current_dft_partition	532
current_instance	534
current_mw_lib	537
current_scenario	539
current_test_mode	541
cut_row	543
date	545
dc_allocate_budgets	547
decrypt_lib	551
define_design_lib	553
define_dft_design	555
define_dft_partition	559
define_libcell_subset	562
define_name_maps	564
define_name_rules	567
define_power_model	580
define_proc_attributes	582
define_routing_rule	587

define_scaling_lib_group	596
define_test_mode	598
define_user_attribute	603
delete_operating_conditions	606
derive_constraints	608
describe_state_transition	610
dft_drc	613
disconnect_net	617
distance	619
drive_of	621
duplicate_logic	623
echo	626
elaborate	628
enable_redirect_bg_commands	633
enable_write_lib_mode	634
encrypt_lib	636
error_info	638
estimate_fp_black_boxes	640
exit	643
extend_mw_layers	645
extract_physical_constraints	647
extract_rc	651
filter	653
filter_collection	656
find	660
find_objects	666
foreach	670
foreach_in_collection	672
generate_mv_constraints	674
generate_rtl_upf	677
get_alternative_lib_cells	679
get_always_on_logic	682
get_app_var	684
get_attribute	687
get_bounds	690
get_buffers	693
get_cells	696
get_clocks	701
get_clusters	704
get_command_option_values	707
get_cross_probing_info	709
get_defined_commands	711
get_design_lib_path	714

get_designs	716
get_dft_hierarchical_pins	719
get_die_area	722
get_dont_touch_cells	724
get_dont_touch_nets	726
get_flat_cells	728
get_flat_nets	732
get_flat_pins	736
get_generated_clocks	740
get_gui_stroke_bindings	743
get_latch_loop_groups	745
get_lib_attribute	747
get_lib_cells	749
get_lib_pins	752
get_libs	755
get_license	758
get_magnet_cells	760
get_matching_nets_for_pattern	762
get_message_ids	765
get_message_info	767
get_multibits	770
get_nets	772
get_object_name	778
get_path_groups	780
get_physical_hierarchy	782
get_pins	783
get_placement_area	786
get_placement_blockages	787
get_polygon_area	790
get_ports	792
get_power_domains	795
get_power_switches	798
get_references	801
get_related_supply_net	804
get_route_zrt_common_options	806
get_route_zrt_global_options	808
get_rp_groups	810
get_scan_cell_names	813
get_scan_cells_of_chain	815
get_scan_chain_names	817
get_scan_chains	819
get_scenarios	821
get_selection	823

get_shift_register_chains	826
get_site_rows	829
get_supply_nets	833
get_supply_ports	836
get_switching_activity	839
get_terminals	842
get_timing_paths	845
get_tracks	852
get_via_rules	856
get_zero_interconnect_delay_mode	859
getenv	860
group	862
group_path	867
group_variable	871
gui_add_annotation	873
gui_bin	878
gui_change_highlight	883
gui_close_window	885
gui_create_attrgroup	887
gui_create_category_rule	889
gui_create_menu	892
gui_create_pref_category	896
gui_create_pref_key	898
gui_create_schematic	901
gui_create_task	903
gui_create_task_item	905
gui_create_tk_palette_type	907
gui_create_tk_task_page	909
gui_create_toolbar	911
gui_create_toolbar_item	913
gui_create_window	915
gui_delete_attrgroup	918
gui_delete_menu	920
gui_delete_toolbar	922
gui_delete_toolbar_item	924
gui_eval_command	926
gui_execute_menu_item	928
gui_exist_pref_category	930
gui_exist_pref_key	931
gui_exist_window	932
gui_get_annotations	934
gui_get_bucket_option	936
gui_get_bucket_option_list	938

gui_get_cell_block_marks	940
gui_get_current_task	942
gui_get_current_task_item	943
gui_get_current_task_page	944
gui_get_current_window	945
gui_get_highlight	948
gui_get_highlight_options	950
gui_get_map_list	952
gui_get_map_option	953
gui_get_map_option_list	955
gui_get_menu_roots	957
gui_get_mouse_tool_option	958
gui_get_pref_keys	960
gui_get_pref_value	961
gui_get_region	963
gui_get_setting	964
gui_get_task_list	966
gui_get_task_page	967
gui_get_toolbar_names	969
gui_get_window_ids	970
gui_get_window_pref_categories	972
gui_get_window_pref_keys	974
gui_get_window_pref_value	976
gui_get_window_types	978
gui_hide_palette	980
gui_hide_toolbar	982
gui_inspect_violations	984
gui_list_attrgroups	986
gui_list_category_rules	988
gui_list_cell_block_marks	991
gui_load_cell_density_mm	993
gui_load_hierarchy_vm	995
gui_load_pin_density_mm	997
gui_load_voltage_area_vm	999
gui_mouse_tool	1000
gui_overlay_layout	1002
gui_query_objects	1004
gui_remove_all_annotations	1007
gui_remove_all_rulers	1009
gui_remove_annotations	1010
gui_remove_category_rules	1012
gui_remove_cell_block_marks	1014
gui_remove_pref_key	1016

gui_remove_ruler	1017
gui_report_hotkeys	1019
gui_report_task	1021
gui_schematic_add_logic	1023
gui_schematic_remove_logic	1025
gui_scroll	1027
gui_select_vmbucket	1029
gui_set_active_window	1031
gui_set_bucket_option	1033
gui_set_cell_block_marks	1035
gui_set_current_task	1037
gui_set_highlight_options	1038
gui_set_hotkey	1040
gui_set_layout_user_command	1043
gui_set_map_option	1045
gui_set_mouse_tool_option	1047
gui_set_pref_value	1049
gui_set_region	1051
gui_set_setting	1053
gui_set_task_list	1055
gui_set_timing_table_paths	1057
gui_set_window_pref_key	1059
gui_show_file_in_editor	1061
gui_show_man_page	1062
gui_show_map	1064
gui_show_palette	1066
gui_show_toolbar	1068
gui_show_url_in_browser	1070
gui_show_window	1071
gui_start	1073
gui_stop	1075
gui_update_attrgroup	1076
gui_update_pref_file	1078
gui_view_port_history	1079
gui_violation_schematic_add_objects	1082
gui_wave_add_signal	1084
gui_write_layout_image	1086
gui_write_window_image	1088
gui_zoom	1090
gui_zoom_all_layouts_to_current_view	1093
help	1094
history	1096
identify_clock_gating	1099

identify_register_banks	1102
if	1107
index_collection	1109
infer_switching_activity	1112
insert_buffer	1115
insert_clock_gating	1120
insert_dft	1123
insert_isolation_cell	1130
insert_mv_cells	1133
is_false	1135
is_true	1137
lib2saif	1139
license_users	1141
link	1143
list	1149
list_attributes	1150
list_commands	1153
list_designs	1155
list_dont_touch_types	1157
list_duplicate_designs	1159
list_files	1161
list_hdl_blocks	1163
list_instances	1165
list_libs	1168
list_licenses	1170
list_size_only_types	1172
list_test_models	1174
list_test_modes	1176
lminus	1178
load_of	1180
load_upf	1182
ls	1186
magnet_placement	1187
man	1190
map_isolation_cell	1192
map_level_shifter_cell	1194
map_power_switch	1196
map_retention_cell	1198
mem	1200
merge_saif	1202
mw_cel_collection	1205
name_format	1207
open_mw_lib	1209

optimize_netlist	1211
optimize_registers	1212
parallel_execute	1217
parse_proc_arguments	1219
preview_dft	1221
print_message_info	1230
print_suppressed_messages	1232
print_variable_group	1234
printenv	1236
printvar	1238
proc_args	1240
proc_body	1242
propagate_constraints	1244
propagate_switching_activity	1250
propagate_user_attributes	1253
push_down_model	1255
pwd	1257
query_cell_instances	1258
query_cell_mapped	1260
query_map_power_switch	1262
query_net_ports	1264
query_objects	1266
query_port_net	1269
query_port_state	1271
query_power_switch	1273
query_pst	1275
query_pst_state	1277
query_qor_snapshot	1279
quit!	1294
quit	1295
read	1296
read_bsd1	1297
read_cell_expansion	1299
read_db	1301
read_ddc	1302
read_file	1304
read_floorplan	1312
read_lib	1314
read_ocvm	1316
read_parasitics	1318
read_partition	1321
read_pin_map	1322
read_saif	1324

read_scan_def	1328
read_sdc	1330
read_sdf	1335
read_sverilog	1338
read_test_model	1339
read_test_protocol	1341
read_verilog	1344
read_vhdl	1346
rebuild_mw_lib	1347
redirect	1349
remove_annotated_check	1354
remove_annotated_delay	1357
remove_annotated_transition	1360
remove_annotations	1362
remove_attribute	1364
remove_auto_path_groups	1366
remove_boundary_cell	1368
remove_boundary_cell_io	1370
remove_bounds	1372
remove_bsd_compliance	1374
remove_bsd_instruction	1376
remove_bsd_linkage_port	1378
remove_bsd_power_up_reset	1380
remove_buffer	1381
remove_buffer_tree	1384
remove_bus	1386
remove_cache	1388
remove_case_analysis	1391
remove_cell	1393
remove_cell_degradation	1395
remove_clock	1397
remove_clock_gating	1399
remove_clock_gating_check	1402
remove_clock_gating_style	1404
remove_clock_groups	1406
remove_clock_latency	1408
remove_clock_sense	1411
remove_clock_transition	1413
remove_clock_uncertainty	1415
remove_congestion_options	1418
remove_constraint	1420
remove_core_area	1422
remove_data_check	1423

remove_design	1426
remove_dft_clock_gating_pin	1429
remove_dft_connect	1430
remove_dft_design	1432
remove_dft_equivalent_signals	1434
remove_dft_location	1435
remove_dft_partition	1437
remove_dft_power_control	1439
remove_dft_signal	1440
remove_die_area	1442
remove_disable_clock_gating_check	1443
remove_disable_timing	1445
remove_dp_int_round	1447
remove_driving_cell	1449
remove_from_collection	1451
remove_from_rp_group	1453
remove_generated_clock	1455
remove_host_options	1457
remove_ideal_latency	1458
remove_ideal_net	1460
remove_ideal_network	1462
remove_ideal_transition	1464
remove_ignored_layers	1466
remove_input_delay	1468
remove_isolate_ports	1471
remove_isolation_cell	1473
remove_keepout_margin	1475
remove_level_shifters	1477
remove_libcell_subset	1478
remove_license	1480
remove_link_library_subset	1482
remove_min_pulse_width	1484
remove_multibit	1486
remove_net	1490
remove_net_routing_layer_constraints	1492
remove_net_search_pattern	1494
remove_net_shape	1496
remove_ocvm	1498
remove_output_delay	1500
remove_pass_directories	1503
remove_path_group	1504
remove_pin_guides	1506
remove_pin_map	1508

remove_pin_name_synonym	1510
remove_placement_blockage	1513
remove_port	1515
remove_power_domain	1517
remove_preferred_routing_direction	1519
remove_propagated_clock	1521
remove_route_guide	1523
remove_routing_rules	1525
remove_rp_group_options	1527
remove_rp_groups	1530
remove_rtl_load	1532
remove_scaling_lib_group	1534
remove_scan_group	1536
remove_scan_link	1538
remove_scan_path	1540
remove_scan_register_type	1542
remove_scan_replacement	1544
remove_scan_skew_group	1546
remove_scan_suppress_toggling	1548
remove_scenario	1549
remove_sdc	1551
remove_sense	1553
remove_target_library_subset	1555
remove_terminal	1557
remove_test_mode	1559
remove_test_model	1560
remove_test_point_element	1561
remove_test_protocol	1563
remove_track	1565
remove_unconnected_ports	1567
remove_upf	1569
remove_user_attribute	1571
remove_user_shape	1573
remove_verification_priority	1575
remove_via	1577
remove_via_ladder_constraints	1579
remove_via_ladder_rules	1581
remove_via_rules	1582
remove_voltage_area	1584
remove_wire_load_min_block_size	1586
remove_wire_load_model	1588
remove_wire_load_selection_group	1590
rename	1592

rename_design	1594
rename_mw_lib	1597
replace_clock_gates	1599
replace_synthetic	1601
report_activity	1603
report_ahfs_options	1606
report_annotated_check	1608
report_annotated_delay	1610
report_annotated_transition	1612
report_app_var	1614
report_area	1616
report_attribute	1621
report_auto_ungroup	1626
report_autofix_configuration	1628
report_autofix_element	1630
report_background_jobs	1632
report_block_abstraction	1634
report_boundary_cell	1637
report_boundary_cell_io	1639
report_bounds	1641
report_bsd_buffers	1643
report_bsd_compliance	1645
report_bsd_instruction	1647
report_bsd_linkage_port	1650
report_bsd_power_up_reset	1652
report_buffer_tree	1654
report_buffer_tree_qor	1656
report_bus	1658
report_cache	1660
report_case_analysis	1665
report_cell	1667
report_cell_mode	1671
report_check_library_options	1673
report_clock	1675
report_clock_gating	1679
report_clock_gating_check	1690
report_clock_timing	1693
report_clock_tree	1704
report_collection	1714
report_compile_options	1718
report_compile_spg_mode	1721
report_congestion	1722
report_congestion_options	1725

report_constraint	1727
report_cross_probing	1735
report_cross_probing_files	1738
report_crpr	1740
report_datapath_gating	1744
report_delay_calculation	1748
report_delay_estimation_options	1754
report_design	1756
report_design_lib	1759
report_design_mismatch	1761
report_dft_clock_controller	1763
report_dft_clock_gating_configuration	1765
report_dft_clock_gating_pin	1767
report_dft_configuration	1769
report_dft_connect	1771
report_dft_design	1772
report_dft_drc_rules	1774
report_dft_equivalent_signals	1776
report_dft_hierarchical_pins	1777
report_dft_insertion_configuration	1779
report_dft_location	1781
report_dft_partition	1783
report_dft_power_control	1785
report_dft_signal	1786
report_direct_power_rail_tie	1789
report_disable_timing	1791
report_dont_touch	1793
report_dp_smartgen_options	1795
report_extraction_options	1797
report_fsm	1799
report_gui_stroke_bindings	1801
report_gui_stroke_builtins	1803
report_hierarchy	1805
report_host_options	1807
report_icc2_options	1808
report_icc_dp_options	1810
report_ideal_network	1812
report_ieee_1500_configuration	1816
report_ignored_layers	1818
report_inbound_cell	1819
report_interclock_relation	1821
report_internal_loads	1824
report_isolate_ports	1826

report_isolation_cell	1828
report_keepout_margin	1831
report_latch_loop_groups	1833
report_level_shifter	1835
report_lib	1839
report_libcell_subset	1856
report_link_library_subset	1858
report_logic_levels	1860
report_logicbist_configuration	1865
report_min_pulse_width	1867
report_missing_constraints	1869
report_mode	1871
report_multibit	1873
report_multibit_banking	1878
report_mv_library_cells	1881
report_mv_qor	1884
report_mw_lib	1889
report_name_rules	1891
report_names	1894
report_net	1897
report_net_fanout	1902
report_net_routing_layer_constraints	1905
report_net_routing_rules	1907
report_net_search_pattern	1910
report_net_search_pattern_delay_estimation_options	1912
report_net_search_pattern_priority	1914
report_ocvm	1916
report_opcond_inference	1921
report_operating_conditions	1923
report_optimize_dft_options	1925
report_partitions	1926
report_pass_data	1928
report_path_budget	1930
report_path_group	1935
report_physical_constraints	1937
report_pin_map	1940
report_pin_name_synonym	1942
report_pipeline_scan_data_configuration	1944
report_port	1946
report_power	1952
report_power_calculation	1960
report_power_domain	1965
report_power_gating	1969

report_power_model	1971
report_power_pin_info	1973
report_power_switch	1977
report_preferred_routing_direction	1979
report_pst	1981
report_qor	1984
report_qtm_model	1989
report_reference	1992
report_resources	1995
report_retention_cell	2001
report_retention_clamp_cell	2004
report_route_zrt_common_options	2007
report_route_zrt_global_options	2008
report_routing_rules	2009
report_rp_group_options	2011
report_saif	2013
report_scaling_lib_group	2017
report_scan_chain	2019
report_scan_compression_configuration	2021
report_scan_configuration	2024
report_scan_group	2028
report_scan_link	2030
report_scan_path	2032
report_scan_register_type	2036
report_scan_replacement	2038
report_scan_skew_group	2040
report_scan_state	2042
report_scan_suppress_toggling	2044
report_scenario_options	2045
report_scenarios	2047
report_self_gating	2050
report_separate_process_options	2053
report_serialize_configuration	2054
report_size_only	2056
report_streaming_compression_configuration	2058
report_supply_net	2060
report_supply_port	2062
report_synlib	2064
report_target_library_subset	2070
report_test_assume	2072
report_test_model	2074
report_test_point_configuration	2076
report_test_point_element	2079

report_testability_configuration	2081
report_threshold_voltage_group	2082
report_timing	2085
report_timing_derate	2097
report_timing_requirements	2101
report_tlu_plus_files	2105
report_top_implementation_options	2107
report_track	2109
report_transitive_fanin	2111
report_transitive_fanout	2113
report_units	2116
report_upf_cell_mismatch	2118
report_use_test_model	2120
report_via_ladder_candidates	2121
report_via_ladder_constraints	2122
report_via_ladder_rules	2124
report_via_rules	2125
report_voltage_area	2127
report_wire_load	2129
report_wrapper_configuration	2132
report_write_lib_mode	2134
reset_autofix_configuration	2135
reset_autofix_element	2137
reset_bsd_configuration	2139
reset_cell_mode	2140
reset_clock_gate_latency	2142
reset_design	2144
reset_dft_clock_controller	2146
reset_dft_clock_gating_configuration	2148
reset_dft_configuration	2149
reset_dft_drc_rules	2150
reset_dft_insertion_configuration	2152
reset_ieee_1500_configuration	2153
reset_logicbist_configuration	2154
reset_mode	2155
reset_path	2157
reset_physical_constraints	2161
reset_pipeline_scan_data_configuration	2162
reset_scan_compression_configuration	2163
reset_scan_configuration	2164
reset_serialize_configuration	2166
reset_streaming_compression_configuration	2167
reset_switching_activity	2168

reset_test_mode	2170
reset_test_point_configuration	2171
reset_testability_configuration	2173
reset_timing_derate	2174
reset_wrapper_configuration	2176
resize_polygon	2178
rewire_clock_gating	2182
rp_group_inclusions	2185
rp_group_instantiations	2187
rp_group_references	2189
run_test_point_analysis	2192
saif_map	2194
save_qtm_model	2201
save_upf	2203
select_block_scenario	2205
set_active_scenarios	2208
set_ahfs_options	2210
set_always_on_cell	2213
set_always_on_strategy	2214
set_analyze_rtl_logic_level_threshold	2216
set_annotated_check	2218
set_annotated_delay	2221
set_annotated_transition	2224
set_app_var	2226
set_aspect_ratio	2228
set_attribute	2230
set_auto_disable_drc_nets	2232
set_auto_ideal_nets	2235
set_autofix_configuration	2236
set_autofix_element	2239
set_balance_registers	2242
set_boundary_cell	2244
set_boundary_cell_io	2250
set_boundary_optimization	2252
set_bsd_ac_port	2254
set_bsd_compliance	2256
set_bsd_configuration	2258
set_bsd_instruction	2262
set_bsd_linkage_port	2267
set_bsd_power_up_reset	2269
set_case_analysis	2271
set_cell_degradation	2273
set_cell_internal_power	2275

set_cell_location	2277
set_cell_mode	2279
set_check_library_options	2281
set_cle_options	2294
set_clock_gate_latency	2296
set_clock_gating_check	2300
set_clock_gating_enable	2303
set_clock_gating_objects	2305
set_clock_gating_registers	2308
set_clock_gating_style	2310
set_clock_groups	2320
set_clock_latency	2323
set_clock_sense	2327
set_clock_transition	2330
set_clock_uncertainty	2332
set_combinational_type	2336
set_compile_directives	2338
set_compile_partitions	2341
set_compile_spg_mode	2344
set_congestion_optimization	2346
set_congestion_options	2348
set_connection_class	2350
set_constant_register_removal	2352
set_context_margin	2354
set_cost_priority	2356
set_critical_range	2358
set_current_command_mode	2360
set_data_check	2362
set_datapath_gating_options	2365
set_datapath_optimization_effort	2369
set_default_drive	2371
set_default_driving_cell	2373
set_default_fanout_load	2376
set_default_input_delay	2378
set_default_load	2380
set_default_output_delay	2382
set_delay_estimation_options	2384
set_design_attributes	2388
set_design_license	2394
set_design_top	2396
set_dft_clock_controller	2397
set_dft_clock_gating_configuration	2401
set_dft_clock_gating_pin	2404

set_dft_configuration	2407
set_dft_connect	2411
set_dft_drc_configuration	2415
set_dft_drc_rules	2418
set_dft_equivalent_signals	2420
set_dft_insertion_configuration	2422
set_dft_location	2425
set_dft_power_control	2429
set_dft_signal	2431
set_direct_power_rail_tie	2442
set_disable_clock_gating_check	2444
set_disable_timing	2446
set_domain_supply_net	2449
set_dont_retime	2451
set_dont_touch	2453
set_dont_touch_network	2456
set_dont_use	2459
set_dp_int_round	2461
set_dp_smartgen_options	2464
set_drive	2471
set_driving_cell	2473
set_dynamic_optimization	2477
set_equal	2479
set_equivalent	2481
set_extraction_options	2483
set_false_path	2485
set_fanout_load	2489
set_fix_hold	2491
set_fix_multiple_port_nets	2493
set_flatten	2496
set_fp_base_gate	2499
set_fsm_encoding	2501
set_fsm_encoding_style	2503
set_fsm_minimize	2505
set_fsm_order	2506
set_fsm_preserve_state	2508
set_fsm_state_vector	2509
set_gui_stroke_binding	2511
set_gui_stroke_preferences	2515
set_host_options	2518
set_hpc_options	2520
set_icc2_options	2522
set_icc_dp_options	2526

set_ideal_latency	2528
set_ideal_net	2530
set_ideal_network	2532
set_ideal_transition	2535
set_ieee_1500_configuration	2537
set_ignored_layers	2539
set_impl_priority	2541
set_implementation	2543
set_input_delay	2545
set_input_transition	2549
set_isolate_ports	2551
set_isolation	2553
set_isolation_cell	2561
set_isolation_control	2563
set_keepout_margin	2566
set_latch_loop_breakers	2569
set_leakage_optimization	2571
set_leakage_power_model	2573
set_level_shifter	2575
set_level_shifter_cell	2581
set_lib_attribute	2584
set_libcell_dimensions	2586
set_libcell_subset	2588
set_libpin_location	2590
set_link_library_subset	2592
set_load	2595
set_local_link_library	2598
set_logic_dc	2600
set_logic_one	2602
set_logic_zero	2604
set_logicbist_configuration	2606
set_map_only	2610
set_max_area	2612
set_max_capacitance	2614
set_max_delay	2617
set_max_fanout	2622
set_max_time_borrow	2624
set_max_transition	2626
set_message_info	2629
set_message_severity	2631
set_min_capacitance	2633
set_min_delay	2635
set_min_library	2639

set_min_pulse_width	2641
set_minimize_tree_delay	2643
set_mode	2645
set_model_drive	2647
set_model_load	2649
set_model_map_effort	2651
set_multi_vth_constraint	2652
set_multibit_options	2655
set_multicycle_path	2658
set_mw_lib_reference	2664
set_mw_technology_file	2666
set_net_routing_layer_constraints	2668
set_net_routing_rule	2670
set_net_search_pattern_delay_estimation_options	2673
set_net_search_pattern_priority	2675
set_opcond_inference	2677
set_operating_conditions	2679
set_opposite	2683
set_optimize_dft_options	2685
set_optimize_registers	2687
set_output_clock_port_type	2691
set_output_delay	2693
set_partial_on_translation	2698
set_path_margin	2700
set_pg_pin_model	2704
set_physical_hierarchy	2706
set_pin_model	2707
set_pin_name_synonym	2709
set_pin_physical_constraints	2711
set_pipeline_scan_data_configuration	2714
set_placement_area	2716
set_port_attributes	2718
set_port_fanout_number	2724
set_port_location	2726
set_port_side	2728
set_power_clock_scaling	2730
set_power_guide	2732
set_power_prediction	2734
set_power_switch_cell	2736
set_prefer	2738
set_preferred_routing_direction	2740
set_preferred_scenario	2742
set_preserve_clock_gate	2744

set_propagated_clock	2747
set_qtm_global_parameter	2749
set_qtm_port_drive	2752
set_qtm_port_load	2754
set_qtm_technology	2756
set_query_rules	2759
set_register_merging	2766
set_register_replication	2768
set_register_type	2771
set_related_supply_net	2774
set_repeater	2776
set_replace_clock_gates	2779
set_resistance	2782
set_resource_allocation	2784
set_retention	2786
set_retention_cell	2791
set_retention_control	2793
set_retention_control_pins	2796
set_retention_elements	2798
set_route_zrt_common_options	2800
set_route_zrt_global_options	2817
set_rp_group_options	2821
set_rtl_load	2827
set_scaling_lib_group	2832
set_scan_compression_configuration	2834
set_scan_configuration	2840
set_scan_element	2846
set_scan_group	2849
set_scan_link	2853
set_scan_path	2855
set_scan_register_type	2864
set_scan_replacement	2866
set_scan_skew_group	2868
set_scan_state	2870
set_scan_suppress_toggling	2872
set_scenario_options	2876
set_scope	2879
set_script_runtime_report_mode	2881
set_self_gating_objects	2883
set_self_gating_options	2885
set_sense	2887
set_separate_process_options	2890
set_serialize_configuration	2892

set_size_only	2895
set_streaming_compression_configuration	2897
set_structure	2901
set_svf	2903
set_switching_activity	2905
set_switching_activity_profile	2910
set_synlib_dont_get_license	2914
set_tap_elements	2916
set_target_library_subset	2917
set_technology	2921
set_test_assume	2923
set_test_point_configuration	2925
set_test_point_element	2928
set_testability_configuration	2932
set_timing_derate	2938
set_timing_ranges	2942
set_tlu_plus_files	2945
set_top_implementation_options	2948
set_transform_for_retiming	2951
set_unconnected	2953
set_ungroup	2955
set_uninitialized_register_value	2957
set_units	2959
set_unloaded_register_removal	2961
set_upf_cell_mismatch	2963
set_upf_query_options	2965
set_user_attribute	2967
set_user_budget	2969
set_utilization	2971
set_verification_priority	2973
set_verification_top	2975
set_via_ladder_candidate	2977
set_via_ladder_constraints	2979
set_via_ladder_rules	2981
set_voltage	2984
set_voltage_model	2987
set_vsdc	2989
set_wire_load	2991
set_wire_load_min_block_size	2996
set_wire_load_mode	2998
set_wire_load_model	3000
set_wire_load_selection_group	3004
set_wrapper_configuration	3006

set_zero_interconnect_delay_mode	3016
setenv	3017
sh	3019
sh_list_key_bindings	3021
shell_is_dcnxt_shell	3023
shell_is_in_exploration_mode	3024
shell_is_in_ndm_mode	3025
shell_is_in_topographical_mode	3026
shell_is_in_xg_mode	3027
sim_assertion_control	3028
sim_corruption_control	3030
sim_replay_control	3032
simplify_constants	3034
size_cell	3036
sizeof_collection	3038
sort_collection	3040
source	3042
split_register_bank	3044
start_icc2	3046
start_icc_dp	3048
streaming_dft_planner	3050
sub_designs_of	3054
sub_instances_of	3057
suppress_message	3060
translate	3062
unalias	3064
ungroup	3065
uniquify	3069
unset_power_guide	3072
unsetenv	3074
unsuppress_message	3076
update_bounds	3078
update_cross_probing_files	3080
update_floorplan	3083
update_lib	3084
update_lib_model	3087
update_lib_pg_pin_model	3089
update_lib_pin_model	3091
update_lib_voltage_model	3093
update_timing	3095
upf_version	3097
use_interface_cell	3099
use_test_model	3101

which	3103
while	3105
win_select_objects	3107
win_set_filter	3109
win_set_select_class	3111
write	3112
write_app_var	3113
write_bsd_rtl	3115
write_bsd	3117
write_cell_expansion	3119
write_collection	3120
write_compile_script	3123
write_def	3126
write_design_lib_paths	3130
write_environment	3132
write_file	3135
write_floorplan	3139
write_icc2_files	3143
write_lib	3146
write_lib_specification_model	3148
write_link_library	3150
write_makefile	3152
write_milkyway	3154
write_multibit_components	3156
write_multibit_guidance_files	3157
write_mw_lib_files	3159
write_parasitics	3161
write_partition	3164
write_partition_constraints	3166
write_physical_constraints	3168
write_qtm_model	3170
write_rp_groups	3172
write_rtl_load	3175
write_saif	3177
write_scan_def	3180
write_script	3182
write_sdc	3186
write_sdf	3190
write_test	3192
write_test_model	3195
write_test_protocol	3197
write_timing_context	3200

acs_check_directories

Checks the Automated Chip Synthesis (ACS) directory structure settings for correctness.

SYNTAX

status **acs_check_directories**
[-verbose *verbosity_level*]

Data Types

verbosity_level integer

ARGUMENTS

-verbose *verbosity_level*

Prints out more messages. Valid values for the *verbosity_level* are **1** to produce verbose output, or **0** for the standard (minimal) output.

DESCRIPTION

The **acs_check_directories** command checks the template that is used to create the Automated Chip Synthesis directory structure, and reports errors, if any are found. This command checks the three variables that comprise the directory structure template:

list_of_file_types, **pass_dependent_file_types**, and **acs_dir**, and searches for the following:

1. Is every file type defined with the proper format in **list_of_file_types**? (The proper format is *file_type(suffix)*, where *suffix* is optional.)
2. Do any two file types have the same name?
3. Are all items in **pass_dependent_file_types** shown in **list_of_file_types**?
4. Are all file types assigned to one path?
5. Is any file type shown in a path not listed in **list_of_file_types**?
6. Is any file type assigned to more than one path?
7. Are any non pass-dependent file types going into a pass-dependent path (a path that contains the pass variable)?

EXAMPLES

The command returns 1 for a correct directory structure setting; otherwise an error message is issued and the command returns 0.

```
prompt> acs_check_directories  
1
```

The following example does not bind the *pass_cstr* file type to any path. This causes problems later when Automated Chip Synthesis attempts to generate constraint files.

```
prompt> acs_check_directories  
Error: file type "pass_cstr" is not assigned to any path.  
0
```

SEE ALSO

```
acs_create_directories(2)  
acs_get_path(2)  
acs_report_directories(2)
```

acs_compile_design

Compiles the constrained RTL to a netlist using constraints propagated from the top-level design.

SYNTAX

```
status acs_compile_design
  [-destination pass_name]
  [-prepare_only]
  [-force]
  [-update]
  [-update_source source_pass]
  design_name
```

Data Types

```
pass_name    string
source_pass  string
design_name  string
```

ARGUMENTS

-destination *pass_name*

Specifies the destination pass directory name for this run. A directory structure for the destination pass is automatically created with the name *pass_name*. All of the generated .ddc files, constraints, scripts, and reports are put under this pass directory.

If you do not specify this option, Automated Chip Synthesis assumes that the pass directory name is @default_pass_name@default_pass_number, where the default pass name is defined by the **acs_default_pass_name** variable, and the default pass number is either 0 (compile_design), or 1 (refine_design).

-prepare_only

Specifies that only the preparation steps like constraint generation, script generation, creation of partition .ddc files, and makefile generation are performed. The actual distributed compile is not done.

-force

Forces the preparation steps such as constraint generation, script generation, creation of partition .ddc files, and makefile generation. By default, if a makefile already exists, the preparation steps are skipped.

-update

Performs an incremental design update by recompiling any compile partitions that have been updated by a previous **acs_merge_design** command, and then performing a top-level boundary compile.

-update_source *source_pass*

Specifies the data directory that contains the merged design, which is the result of the **acs_merge_design** command. The **acs_compile_design -update** command loads the designs from *source_pass/db/pre_compile* before performing the incremental design update.

If this option is not specified, but the **-update** option is specified, Automated Chip Synthesis looks for the design in memory. If the design is not found in memory, Automated Chip Synthesis looks in the default directory, `${acs_default_pass_name}0_incr/db/pre_compile`.

design_name

Specifies the name of the top-level design.

DESCRIPTION

The **acs_compile_design** command performs an initial mapping from RTL to gates using parallel compile strategy. You must provide a constrained GTECH design in memory or as a .ddc file in the `${acs_work_dir}/elab/db` directory. By default, **acs_compile_design** partitions the design at the first level of the logical hierarchy if no user partitions are set. The top-level constraints are propagated to the partitions, the compile scripts are generated, and the makefile is written out. A parallel compile is performed using LSF or the gmake utility, according to the settings of the **acs_num_parallel_jobs** variable.

Before running this command, review the default settings in the `.synopsys_dc.setup` file located in your project directory.

This command is implemented as a Tcl procedure that is visible and customizable. For details about how to change the behavior of this procedure, see the *Automated Chip Synthesis User Guide*.

EXAMPLE

The following example shows how to perform an incremental design update. Assume that you have compiled your design using **acs_compile_design**. Later, you modified one subdesign named *desA* and created a new design database in `${acs_work_dir}/elab/db` with the changed subdesign *desA*. Assume that you are using the default directories. Then you can merge the changed design with an existing mapped design database of a previous compile run and recompile the changes by using the following commands:

```
prompt> acs_merge_design top_design_name \  
-update desA -des merged_pass
```

```
prompt> acs_compile_design top_design_name \  
-update -update_source merged_pass
```

SEE ALSO

acs_merge_design(2)
acs_recompile_design(2)
acs_refine_design(2)

acs_create_directories

Creates the project directory tree for Automated Chip Synthesis.

SYNTAX

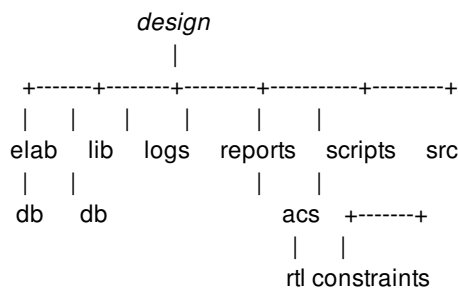
int **acs_create_directories**

ARGUMENTS

This command has no arguments.

DESCRIPTION

Creates the pass-independent directories of the project directory tree for Automated Chip Synthesis. This command creates the default directory structure shown below:



With the exception of the `elab/db` directory, all directories shown above are always created. The `elab/db` directory contains the GTECH design database used as input to the **`acs_compile_design`** command. You can change the location for the GTECH design database by defining a new location for the `elab_ddc` file type. Redefine file locations by setting the **`acs_user_dir`** variable. For more information about Automated Chip Synthesis file types and customizing the directory structure, see the **`acs_user_dir`** command man page.

If you define additional pass-independent directories by setting the **acs_user_dir** variable, the **acs_create_directories** command also creates these directories.

The **acs_create_directories** command does not create the pass-dependent directories. Automated Chip Synthesis automatically creates these directories when you run the pass commands. You can manually create pass-dependent directories using the **create_pass_directories** command.

The @ character cannot be used as a literal character in Automated Chip Synthesis directory paths. This character is reserved for

specifying the names of pass-dependent directories.

EXAMPLES

The following command creates the project directory tree for Automated Chip Synthesis.

```
prompt> acs_create_directories  
Start creating ACS directory structure  
Creating directory /user/my_design/scripts/acs...  
Creating directory /user/my_design/reports...  
Creating directory /user/my_design/src/constraints...  
Creating directory /user/my_design/src/rtl...  
Creating directory /user/my_design/elab/db...  
Creating directory /user/my_design/lib/db...  
Creating directory /user/my_design/logs...
```

SEE ALSO

`acs_get_path(2)`
`acs_report_directories(2)`
`create_pass_directories(2)`

acs_customize_directory_structure

This command is obsolete. See the **acs_user_dir** variable for information about customizing the Automated Chip Synthesis directory structure.

acs_get_parent_partition

Creates a collection of designs that are compiled partitions containing the specified subdesign.

SYNTAX

```
string acs_get_parent_partition  
    design_name  
    [-hierarchy]  
    [-list]
```

Data Types

design_name string

ARGUMENTS

design_name

Specifies the name of a subdesign.

-hierarchy

Causes the command to collect all parent partitions of the specified design. By default only the immediate parent partition is returned.

-list

Causes the command to return a list of names. Without this option a collection of design objects is returned.

DESCRIPTION

The **acs_get_parent_partition** command returns the lowest-level compile partition that contains the specified subdesign. If the specified subdesign is a compile partition, the command returns the subdesign. If the **-hierarchy** option is specified, all partitions that contain the subdesign are included in the collection. If the subdesign is in the subtree of a multiply instantiated design, all parent partitions of all instances of this multiply instantiated design are included in the result. The command creates a collection containing the design objects, unless the **-list** option is specified. In this case, the command returns a Tcl list with the names of the designs.

SEE ALSO

`set_compile_partitions(2)`

acs_get_path

Gets the path location for the specified file. To specify a file, you specify its file type and for pass-dependent files, its pass directory.

SYNTAX

```
string acs_get_path  
-file_type filetype  
[-mode read | write]  
[-pass pass_name]  
[-name filename [-append]]  
[-relative]
```

Data Types

```
filetype    string  
pass_name   string  
filename    string
```

ARGUMENTS

-file_type *filetype*

Specifies the file type. The **acs_get_path** command returns the path location that contains the specified file. The file type must be one of the Automated Chip Synthesis file types. For information about file types, see the *Automated Chip Synthesis User Guide*.

-mode read | write

Specifies whether the command runs in read mode or write mode. In either mode, the command returns the directory name for the specified file. In write mode, in addition to returning the directory name, the **acs_get_path** command creates the directory if it does not exist.

By default, the command runs in read mode.

-pass *pass_name*

Specifies the pass name for pass-dependent files. In read mode only, you can request the generic path for a pass-dependent file type by specifying the pass variable (@pass@) for the *-pass_name* value.

The @ character cannot be used as a literal character in Automated Chip Synthesis. This character is reserved for specifying the names of pass-dependent directories.

-name *filename*

Appends the specified file name to the end of the returned path. You must specify the full name of the file if you don't specify the **-append** option. If the **-append** option is specified, only the base name of the file is needed.

By default, only the directory is returned.

-append

Appends the pass name, file type, and the default file suffix to the base name of the file name specified by the **-name** option, if necessary to create unique file names for pass-dependent files located in pass-independent directories.

By default only the specified file name (plus the default file suffix) is returned.

You must use this option with the **-name** option. If you specify **-append** and do not specify the **-name** option, an error occurs.

-relative

Returns the directory location relative to the Automated Chip Synthesis project directory (as specified by the **acs_work_dir** variable).

By default, the **acs_get_path** command returns the absolute path.

DESCRIPTION

The **acs_get_path** command locates the files used by Automated Chip Synthesis.

Because you can customize the Automated Chip Synthesis directory structure, Automated Chip Synthesis must provide a dynamic method for locating files. The **acs_get_path** command provides this capability. In general, you use this command when writing custom scripts for Automated Chip Synthesis, instead of hardcoding file locations. This ensures that your custom scripts work with both the default and customized directory structures.

EXAMPLES

The following examples are based on the default directory setting. The output may vary for non-default directory structures.

The following command returns the absolute path for the postcompile .ddc file in the PASSX directory tree:

```
prompt> acs_get_path -file_type post_ddc -pass PASSX  
/usr/my_design/PASSX/db/post_compile
```

The following command returns the path for the same file, relative to the project root directory, /usr/my_design:

```
prompt> acs_get_path -file_type post_ddc -pass PASSX -relative  
PASSX/db/post_compile
```

The following command returns the path for the same file, including the specified file name:

```
prompt> acs_get_path -file_type post_ddc -pass PASSX -name my_design  
/usr/my_design/PASSX/db/post_compile/my_design.ddc
```

The following command returns the generic path for the postcompile .ddc files:

```
prompt> acs_get_path -file_type post_ddc -pass @pass@  
/usr/my_design/${pass}/db/post_compile
```

Assume that the directory structure is customized so that the precompile and postcompile .ddc files for each pass are placed in the \$acs_work_dir/db directory. In this case, Automated Chip Synthesis appends the file type and pass name to make each file name

unique. The following command returns the unique file name used by Automated Chip Synthesis.

```
prompt> acs_get_path -file_type post_ddc -pass test \  
? -name my_design -append  
/usr/my_design/db/my_design.test.post_ddc.ddc
```

SEE ALSO

- acs_create_directories(2)
- acs_report_directories(2)
- create_pass_directories(2)
- acs_work_dir(3)

acs_merge_design

Preprocesses a design for incremental design update by merging the modified designs and their parent compile partitions with the mapped design being updated.

SYNTAX

```
status acs_merge_design
  [-auto_update | -update design_list]
  [-unmapped source_dir]
  [-mapped data_dir]
  [-type pre | post]
  [-destination dest_dir]
  [-reference pre_compile_partition_dir]
  design_name
```

Data Types

<i>design_list</i>	string
<i>source_dir</i>	string
<i>data_dir</i>	string
<i>dest_dir</i>	string
<i>pre_compile_partition_dir</i>	string
<i>design_name</i>	string

ARGUMENTS

-auto_update

Automatically determines the designs to be updated using the information in timestamp tables written by the **acs_read_hdl** -
[auto_]update command.

-update *design_list*

Specifies the designs to be updated in the design already mapped.

-unmapped *source_dir*

Specifies the directory with the .ddc files of the design database that is the source for the modified designs. From this design database, the **acs_merge_design** command fetches the modified designs specified with the **-update** option and all parent partitions of these designs upwards to the top-level hierarchy. These designs are usually RTL/unmapped (GTECH) designs, but they can also be mapped designs.

All .ddc files in this directory must comprise a complete design database for the top-level design. The designs may be arbitrarily distributed among the .ddc files.

If you do not specify this option, **acs_merge_design** picks up the .ddc files from the elab_ddc directory. The default location of this directory is \$acs_work_dir/elab/db.

-mapped data_dir

Specifies the data directory that contains the mapped .ddc files of the top-level design and its compile partitions. The directory must be an Automated Chip Synthesis data directory and the .ddc files located under the subdirectory must contain partition and constraints information.

If you do not specify the **-type** option, **acs_merge_design** reads the mapped design from the post_ddc directory. The default location of this directory is \$acs_work_dir/data_dir/db/post_compile.

If you specify the **-type pre** option, the **acs_merge_design** command reads the mapped design from the pre_ddc directory. The default location of this directory is \$acs_work_dir/data_dir/db/pre_compile.

If you do not specify this option or the **-type** option, **acs_merge_design** reads the mapped design from the post_ddc directory in the default data directory. The default data directory is \${default_pass_name}0, where the default pass name is defined by the **acs_default_pass_name** variable.

If you do not specify this option, but you do specify the **-type pre** option, **acs_merge_design** reads the mapped design from the pre_ddc directory in the default data directory.

-type pre | post

Specifies whether the mapped design is read from the pre_ddc directory or the post_ddc directory of the mapped data directory. (See the description of the **-mapped** option for more information about the mapped data directory).

If you do not specify this option, the mapped design is read from the post_ddc directory.

-destination dest_dir

Specifies the name of the destination directory for this run. The **acs_merge_design** command automatically creates the directory structure for the destination directory. All of the generated .ddc files are put in the pre_ddc directory in the destination directory.

If you do not specify this option, **acs_merge_design** uses a default destination directory named \${default_pass_name}0_incr, where the default pass name is defined by the **acs_default_pass_name** variable.

-reference pre_compile_partition_dir

Specifies the directory in which the pre_compile_partitions are found. The earliest timestamp of the precompile partitions is used as the time at which the last compile command was issued. Any design with a later timestamp is selected for update when the **-auto_update** option is specified. The default location is pass0/db/pre_compile.

design_name

Specifies the name of the top-level design.

DESCRIPTION

The **acs_merge_design** command loads the mapped design from the mapped data directory (see the description of the **-mapped** option), overrides the updated designs and all its parent partitions up to the top level in the hierarchy with the GTECH implementations from the unmapped source directory. (See the description of the **-unmapped** option). For any updated design the parent partitions up to the top level are also replaced in order to ensure correct design functionality, since these partitions may have been optimized using design properties of the old version of the updated designs, such as logical constants. Finally, the merged design is saved in the destination directory. (See the description of the **-destination** option).

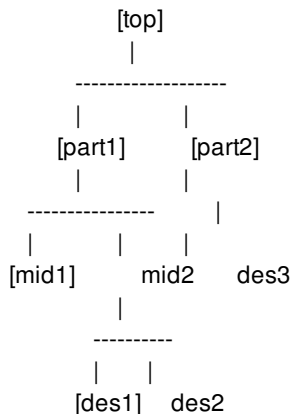
The **acs_merge_design** command marks all partitions that are replaced as being updated, so that a subsequent Automated Chip

Synthesis **compile** command automatically detects the updated designs for recompile when using the **-update** option.

The **-auto_update** option can be used to automatically determine the designs to replace. This is done by comparing the timestamp of the designs in the unmapped design (as recorded in the timestamp tables created by the **acs_read_hdl -[auto_update]** command) and the timestamp of the precompile partitions. If a design has a later timestamp than the earliest timestamp of the precompile partition, then it is placed in the list of updated designs.

EXAMPLES

The following is an example of the **acs_merge_design** command. Assume you have the following design:



In this example, the designs with brackets ([]) are Automated Chip Synthesis compile partitions. The designs without brackets are non-partitions. The compile partition named part1 is the parent partition of des2.

When you perform an initial Automated Chip Synthesis compile, the tool takes the unmapped RTL design from the elab/db directory or from memory, partitions the design, places the GTECH .ddc files of the unmapped partitions in the \$acs_work_dir/pass0/db/pre_compile directory, and the .ddc files for the mapped design in the \$acs_work_dir/pass0/db/post_compile directory. In these directories, each partition in the design has a corresponding .ddc file whose name matches the partition's name.

Assume that you modify the des2 subdesign and put the new RTL design database with the changed design into the elab/db directory. Then you merge your changes in des2 into the design that is already mapped from the initial Automated Chip Synthesis compile, as shown in the following example:

```
prompt> acs_merge_design top -update des2 \
        -unmapped elab/db -mapped pass0
```

The **-unmapped** and **-mapped** options are not necessary, since **acs_merge_design** picks up the modified designs from elab/db and merges them with \$acs_work_dir/pass0/db/post_compile by default. The tool picks up the des2 design and its parent partitions named part1 and top containing designs des2, mid2, part1, and top from the directory specified by the **-unmapped** option (elab/db in this example) and merges them with the design in the directory specified by the **-mapped** option (\$acs_work_dir/pass0/db/post_compile in this example). The result of this merge is placed by default in the \$acs_work_dir/pass0/incr/db/pre_compile directory.

SEE ALSO

acs_compile_design(2)

acs_read_hdl(2)
acs_recompile_design(2)
acs_refine_design(2)

acs_read_hdl

Reads in the HDL source code of a design and generates the GTECH representation.

SYNTAX

```
status acs_read_hdl
  [design_name]
  [-hdl_source file_or_dir_list]
  [-exclude_list file_or_dir_list]
  [-format {verilog | vhdl}]
  [-recurse]
  [-no_dependency_check]
  [-no_elaborate]
  [-library design_lib_name]
  [-verbose]
  [-auto_update | -update file_list]
  [-destination destination_dir]
  [-sv_package_files file_list]
```

Data Types

```
design_name      string
file_or_dir_list list
design_lib_name  string
file_list       list
destination_dir string
```

ARGUMENTS

design_name

Specifies the name of the top-level design. Elaboration starts at the specified module and elaborates the entire design subtree under that module. This argument is required unless the **-no_elaborate** option is specified.

-hdl_source *file_or_dir_list*

Specifies a list of files or directories containing the source code of the design. The **acs_read_hdl** command analyzes the contents of all files in the list (and in the subdirectories if the **-recurse** option is set) according to the **acs_verilog_extensions**, **acs_vhdl_extensions**, **acs_exclude_list**, and **acs_exclude_extensions** variables.

If this option is not specified, the command uses the **acs_hdl_source** Tcl variable by default. If the **acs_hdl_source** variable is not defined, the **search_path** variable is used. The command-line argument overrides the variables. The names in the **-hdl_source** lists can contain wildcards. The command performs a Tcl glob-style expansion on each list item.

-exclude_list *file_or_dir_list*

Specifies a list of files and directories that are not to be analyzed. If the command expands an item from the **-hdl_source** list, it checks the file or directory against all entries of this list and ignores the file or directory if it is covered by at least one of the entries.

If this option is omitted, the command uses the **acs_exclude_list** Tcl variable. The command-line argument overrides the variable. It is possible to apply Tcl's UNIX-like filename globbing features by using the *****, **?**, and **[]** in file and directory names in this list. Add a file name (wildcards are allowed) without a path to exclude all files with that name regardless of their location.

-format {verilog | vhdl}

Specifies that **acs_read_hdl** is to only look for files of the specified language with extensions from the corresponding extensions list for the **acs_verilog_extensions** or **acs_vhdl_extensions** variables. The default is that the command reads in all files it finds with Verilog extensions and VHDL extensions.

Use this option when you want to specify file names in the **-hdl_source** list that do not have one of the language-specific extensions.

-recurse

Specifies that the command is to look into subdirectories of directories specified in the **-hdl_source** list, and recursively collect source files from that location.

The default is that the command looks into the specified directories, but does not look in their subdirectories.

-no_dependency_check

Causes the command not to detect Verilog include files, not to determine VHDL libraries, and not to reorder the source file list. Instead, the command processes the HDL source file list from left to right and analyzes VHDL files into the current working library or the library specified with the **-library** option.

-no_elaborate

Prevents **acs_read_hdl** from performing the elaboration. The command stops after all source files are analyzed. Specify this option when you want to manually elaborate the design after the command finishes.

-library *design_lib_name*

Specifies the library that is used as the working library for all VHDL files.

-verbose

Prints out more messages.

-auto_update

Automatically determines the HDL files to update, puts them in order according to the dependency (unless the **-no_dependency_check** option is specified) and analyzes and elaborates the files. The designs defined in the specified HDL file(s) replace the corresponding designs in the unmapped design in the destination directory. The option also creates the tables of timestamps used by the subsequent **acs_read_hdl -auto_update**, **acs_read_hdl -update**, and **acs_merge_design -auto_update** commands.

This option is mutually exclusive with the **-update** option.

-update *file_list*

Analyzes and elaborates the specified HDL source files in the given order. The designs defined in the specified HDL file(s) replace the corresponding designs in the unmapped design in the destination directory. The option also creates the tables of timestamps used by the subsequent **acs_read_hdl -auto_update**, **acs_read_hdl -update**, and **acs_merge_design -auto_update** commands.

This option is mutually exclusive with the **-auto_update** option.

-destination *destination_dir*

Specifies the directory to which the unmapped design is written when either the **-update** or the **-auto_update** option is specified. The default is that the command writes the unmapped design and the timestamp tables to the elab/db directory.

DESCRIPTION

The **acs_read_hdl** command reads in the HDL source files of a design, analyzes them, and elaborates the design starting at the specified top-level module. It retains the resulting GTECH representation in memory.

To locate the source files, the command expands each item from the **-hdl_source** list. The list is either specified as an option with **-hdl_source** at the command line, or set as the value of the **acs_hdl_source** Tcl variable. If neither of these methods are used, then the **search_path** variable is used. The **acs_read_hdl** command performs a Tcl glob-style expansion on an item and checks if the result is excluded by the **acs_exclude_list** or the **acs_exclude_extensions** variable. If it is not excluded and a file ends with one of the extensions from the **acs_vhdl_extensions** variable extensions list, it is considered a VHDL source file. If the file ends with one of the **acs_verilog_extensions**, it is considered a Verilog source file.

If an expansion is a directory, the command collects the files from the directory, and recursively from its subdirectories if the **-recursive** option is set. The command then performs all extension checks on these files. If the **-format** option is set, only files with Verilog or VHDL extensions are collected (based on the value of the option).

After **acs_read_hdl** collects all of the source files, it performs the following dependency checks (unless the **-no_dependency_check** option is specified).

- Detects Verilog include files to verify that they are not analyzed as source code files.
- If a Verilog include file that appears only in the list of HDL source files is detected, but its directory does not appear in the global **search_path** variable, and it is included in the Verilog source code without giving the path, then **acs_read_hdl** appends the include file's directory to the **search_path**. This enables the **analyze** or **read_verilog** commands to locate the file. You are notified about the changes made.
- Detects Verilog macro usage and definition and reorders files accordingly to ensure that they are analyzed in the correct order. This might not always be possible, such as when a macro is defined several times in different files or when macros are defined in files included by other include files.
- Determines the target library for each VHDL file. However, if the **-library** option is specified, this step is skipped and all VHDL files are analyzed into the specified design library.
- Orders the VHDL source files to ensure that they are analyzed in the correct order.

After analyzing all source files, the design is elaborated from the top-level module, unless the **-no_elaborate** option is specified. The resulting GTECH representation is retained in memory.

If either the **-update** or the **-auto_update** option is specified, then the GTECH representation (unmapped design) is written to the specified destination directory. The default destination directory is elab/db.

When the **acs_read_hdl -auto_update** or **acs_read_hdl -update** command is called the first time, all of the source HDL files are analyzed and elaborated to create the unmapped design and the timestamp tables. Subsequent **acs_read_hdl -auto_update** commands only analyze and elaborate the updated HDL source files and subsequent **acs_read_hdl -update** commands only analyze and elaborate the specified HDL source files. The updated designs are replaced in the unmapped design. The first issuance of the **acs_read_hdl** command is determined by the presence of the unmapped design file and timestamp tables in the destination directory.

EXAMPLES

The following example assumes that the current directory is the source directory. It specifies the source file list at the command line and calls the command with the name of the top-level entity.

```
prompt> acs_read_hdl -hdl_source {.} -recurse E1
```

The next example specifies extensions for Verilog files that are different than the default ({.v}), sets the **-hdl_source** list and the **exclude_list** list, and calls the command with the name of the top-level module name, forcing it to only collect files with Verilog extensions.

```
prompt> set acs_verilog_extensions { .ve .VE }  
prompt> set acs_hdl_source {mod1/src mod2/*/src}  
prompt> set acs_exclude_list {mod1/src/incl mod2/*/src/incl}  
prompt> acs_read_hdl -f verilog -no_dep TOP
```

Note that excluding include directories explicitly is only necessary if include files have the same extensions as source files and not all include files are included in the source.

```
prompt> acs_read_hdl -hdl_source src/mixed -auto_update
```

The first time the above command is issued, all of the mixed (Verilog and VHDL) HDL files from the specified directory are analyzed and elaborated, and the resulting unmapped design is written to the default destination directory along with the timestamp tables. The timestamp tables are used by the **acs_merge_design -auto_update** command to determine which designs are new, so that the partitions which depend on the updated designs are recompiled by the **acs_compile_design -update** command. The subsequent calls to the above command only analyze and elaborate the updated HDL files, and the corresponding designs are replaced in the unmapped design in the destination directory.

SEE ALSO

- acs_merge_design(2)
- acs_compile_design(2)
- analyze(2)
- elaborate(2)
- search_path(3)

acs_recompile_design

Compiles an unmapped constrained .ddc file using time budgets. The time budgets are created by using a previously mapped design.

SYNTAX

```
status acs_recompile_design
  -budget_source budget_pass
  -destination destination_pass
  [-source source_pass]
  [-prepare_only]
  [-force]
  [-update]
  [-update_source source_pass]
  design_name
```

Data Types

<i>budget_pass</i>	string
<i>destination_pass</i>	string
<i>source_pass</i>	string
<i>design_name</i>	string

ARGUMENTS

-budget_source *budget_pass*

Specifies a pass directory from which the mapped design is picked up to generate time budgets for this pass.

-destination *destination_pass*

Specifies the pass name for this run. A directory structure for the pass is automatically created with the name *destination_pass*. All of the generated .ddc files, constraints, scripts, and reports are put under this pass directory.

-source *source_pass*

Specifies the pass directory from which Automated Chip Synthesis gets unmapped designs. The unmapped design is mapped using the budgeting information provided by the design from *budget_pass*. If this option is not used, Automated Chip Synthesis looks for the design in the memory. If the design is not in memory, the tool looks for the design from the pass0, which is the default source pass.

-prepare_only

Specifies that only preparation steps like constraint generation, script generation, creation of partition .ddc files, and makefile generation are performed. The actual distributed compile is not done.

-force

Specifies to force preparation steps such as constraint generation, script generation, creation of partition .ddc files, and makefile generation. By default, if a makefile already exists, the preparation steps are skipped.

-update

Performs an incremental design update by recompiling any compile partitions that have been updated by a previous **acs_merge_design** command, and then doing a top-level boundary compile.

-update_source source_pass

Specifies a pass directory that contains the merged design (the result of the **acs_merge_design** command) with the updated designs to be recompiled. If this option is not specified while the **-update** option is specified, then it looks for the design in memory. If the design is not in memory, then the tool looks for the default directory `${acs_default_pass_name}1_incr/db/pre_compile`.

design_name

Specifies the name of the top-level design. This argument is required.

DESCRIPTION

The **acs_recompile_design** command performs mapping from RTL to gates using parallel compile strategy. You must provide a constrained GTECH design in memory or specify an appropriate GTECH pass directory using the **-source** option. The command uses the partition information from the unmapped source design. The constraints for the partitions are derived by performing time budgeting on the mapped design coming from the *budget_pass*. The compile scripts are generated, and the makefile is written out. A parallel compile is then performed using LSF or the gmake utility, according to the settings of the **acs_num_parallel_jobs** variables.

Before you run this command, be sure to review the default settings in the `.synopsys_dc.setup` file in your project directory.

This command is implemented as a Tcl procedure that is visible and customizable. For details about how to change the behavior of this procedure, see the *Automated Chip Synthesis User Guide*.

EXAMPLES

The following example shows how to compile the design and rebudget the mapped netlist using **acs_refine_design**:

```
prompt> acs_compile_design DESIGN
```

```
prompt> acs_refine_design DESIGN
```

In the example above, **acs_compile_design** saves the compiled design in the `pass0` directory. The **acs_refine_design** command reads the mapped design netlist from the `pass0` directory, performs budgeting on the mapped netlist and saves the refined design in the `pass1` directory.

The following example incrementally refines the design that has been updated by the previous **acs_merge_design** command.

```
prompt> acs_merge_design -auto_update -dest merged_pass DESIGN
```

```
prompt> acs_compile_design -update -update_source merged_pass \
```

-dest compiled_pass DESIGN

```
prompt> acs_refine_design -source compiled_pass \  
         -destination refined_pass DESIGN
```

The **acs_merge_design** command automatically identifies subdesigns that have been updated and creates a new design database in merged_pass. The subsequent **acs_compile_design** performs compile only on partitions that are marked as changed by **acs_merge_design** and saves the database in compiled pass. Finally, **acs_refine_design** reads the mapped design database from compiled_pass and performs budgeting on the updated design.

SEE ALSO

acs_compile_design(2)
acs_merge_design(2)
acs_refine_design(2)

acs_refine_design

Refines an already mapped design.

SYNTAX

```
status acs_refine_design
  [-source pass_name]
  [-destination pass_name]
  [-prepare_only]
  [-force]
  [-update]
  [-update_source source_pass]
  design_name
```

Data Types

```
pass_name    string
source_pass  string
design_name  string
```

ARGUMENTS

-source *pass_name*

Specifies the Automated Chip Synthesis pass from which to get the mapped design.

If you do not specify this option, Automated Chip Synthesis assumes that the pass directory name is @pass@0, where string @pass@ is defined by the **acs_default_pass_name** variable.

If you do not specify a source pass, Automated Chip Synthesis uses pass0.

-destination *pass_name*

Specifies the pass name for this run. A directory structure for the pass is automatically created with the name *pass_name*. All of the generated .ddc files, constraints, scripts, and reports are put in this pass directory.

If you do not specify this option, Automated Chip Synthesis assumes that the pass directory name is @pass@1, where the string @pass@ is defined by the **acs_default_pass_name** variable.

-prepare_only

Only the preparation steps such as constraint generation, script generation, creation of partition .ddc files, and makefile generation are performed. The actual distributed compile is not done.

-force

Forces the preparation steps such as constraint generation, script generation, creation of partition .ddc files, and makefile generation. By default, if a makefile already exists, the preparation steps are skipped.

-update

Performs an incremental design update by refining any compile partitions that have been updated by a previous **acs_merge_design** command, and then doing a top-level boundary compile.

-update_source *source_pass*

Specifies a pass directory which contains the merged design (result of **acs_merge_design**) with the updated designs to be refined. If this option is not specified when the **-update** option is specified, then the tool looks for the design in memory. If the design is not in memory, the tool looks for the default directory `${acs_default_pass_name}1_incr/db/pre_compile`.

design_name

The name of the top-level design. This argument is required.

DESCRIPTION

The **acs_refine_design** command remaps a design that is already mapped coming from the source pass directory specified. The partitioning of the source pass is preserved. The constraints for the partitions are derived by performing time budgeting on the mapped design coming from the source pass. The compile scripts are generated, and the makefile is written out. A parallel incremental or top-level compile is then performed using LSF or the gmake utility according to the settings of the **acs_num_parallel_jobs** variables.

Before you run this command, review the default settings in the `.synopsys_dc.setup` file located in your project directory.

This command is implemented as a Tcl procedure that is visible and customizable. For details of how to change the behavior of this procedure, refer to the *Automated Chip Synthesis User Guide*.

SEE ALSO

`acs_compile_design(2)`
`acs_merge_design(2)`
`acs_recompile_design(2)`

acs_remove_dont_touch

Removes the dont_touch attributes on the specified designs.

SYNTAX

```
status acs_remove_dont_touch  
[-force | design_list]
```

Data Types

design_list list

ARGUMENTS

-force

Removes the dont_touch attributes on all compile partitions, including multiply-instantiated designs.

This option and the *design_list* argument are mutually exclusive.

design_list

Removes the dont_touch attributes on the specified designs.

This argument and the **-force** option are mutually exclusive.

DESCRIPTION

The **acs_remove_dont_touch** command removes the dont_touch attributes on designs according to the options. If you do not specify any options, the command removes the dont_touch attributes on all compile partitions except the multiply-instantiated designs.

This command applies to compile partitions only. It cannot remove the dont_touch attributes from nonpartition designs. For detailed information about partitions, see the man pages for the **set_compile_partitions** and **report_partitions** commands.

SEE ALSO

get_attribute(2)

```
report_partitions(2)
set_attribute(2)
set_compile_partitions(2)
set_dont_touch(2)
```

acs_report_attribute

Reports the ACS attributes on the specified partitions.

SYNTAX

```
status acs_report_attribute  
  [-partitions partition_list]  
  [attribute_name]
```

Data Types

```
partition_list  list  
attribute_name string
```

ARGUMENTS

-partitions *partition_list*

Specifies the compile partitions whose ACS attributes are reported.

If you do not specify this option, the command reports all compile partitions that have the specified attribute.

attribute_name

Specifies the attribute to report.

If you do not specify this option, the command reports all attributes set on the specified compile partitions.

DESCRIPTION

The **acs_report_attribute** command reports the ACS attributes set on the specified compile partitions.

SEE ALSO

```
acs_set_attribute(2)  
set_compile_partitions(2)  
report_partitions(2)
```

acs_report_directories

Reports the current directory structure settings.

SYNTAX

```
status acs_report_directories
[-file_types type_list]
```

Data Types

type_list list

ARGUMENTS

-file_types *type_list*

The list of file types you would like to be reported.

DESCRIPTION

Returns all the paths that are defined in the directory structure template. Currently, only elements in `acs_dir` are printed.

When `-file_types` option is not used, `acs_report_directories` returns all the paths that are defined in the directory structure template and the file types bound to each path. If a list of file types is given with `-file_types` option, the command prints out the path where each given file type is assigned to.

EXAMPLES

Example 1:

```
prompt> acs_report_directories
Report of ACS directory structure
/remote/user/acs/new/@pass@: { makefile oc.tcl cstr.tcl }
/remote/user/acs/new/@pass@/constraints: { pass_cstr }
/remote/user/acs/new/@pass@/db/post_compile: { post_ddc }
/remote/user/acs/new/@pass@/db/pre_compile: { pre_ddc }
...
```


Example 2:

```
prompt> acs_report_directories \  
-file_type {pass_compile_script pass_cstr elab_ddc}  
Report of ACS directory structure  
pass_compile_script -> /remote/dtg215/rwang/acs/new/@pass@/scripts  
pass_cstr -> /remote/dtg215/rwang/acs/new/@pass@/constraints  
elab_ddc -> /remote/dtg215/rwang/acs/new/elab/db
```

SEE ALSO

acs_create_directories(2)
acs_check_directories(2)
acs_get_path(2)

acs_report_user_messages

Reports the number of error, warning, and information messages.

SYNTAX

```
status acs_report_user_messages  
  [-total]  
  [-errors]  
  [-warnings]  
  [-infos]  
  [-reset]
```

ARGUMENTS

-total

Prints the total number of error messages that have occurred since the the current session began. The default is to print the number of error messages since the last call to the **acs_report_user_messages** command.

-errors

Prints the number of error messages that have occurred since the last call to **acs_report_user_messages**.

-warnings

Prints the number of warning messages that have occurred since the last call to **acs_report_user_messages**.

-infos

Prints the number of informational messages that have occurred since the last call to **acs_report_user_messages**.

-reset

Resets the counters for the errors, warnings, and informational messages. However, subsequent calls to **acs_report_user_messages** with the **-total** option prints the number of error, warning, and information messages since the start of the session.

DESCRIPTION

The **acs_report_user_messages** command reports the number of error, warning, and information messages.

You can use any combination of the **-errors**, **-warnings**, and **-infos** options with the command. If no options are specified, then the effect is the same as if all three of the options are specified; all three types of messages are printed. Note that the **-total** option is

independent of these three options.

The ACS chip-level compile commands **acs_compile_design**, **acs_recompile_design**, and **acs_refine_design** issue the **acs_report_user_messages -reset** command when the commands begin.

EXAMPLES

The following example prints the number of error, warning, and information messages since the last call to **acs_report_user_messages**:

```
prompt> acs_report_user_messages
```

The following example prints the number of warning and error messages that have occurred since the last call to **acs_report_user_messages**:

```
prompt> acs_report_user_message -errors -warnings
```

The following example prints the number of errors since the start of the session:

```
prompt> acs_report_user_message -errors -total
```

SEE ALSO

acs_compile_design(2)
acs_recompile_design(2)
acs_refine_design(2)

acs_reset_directory_structure

Resets the project directory tree of Automated Chip Synthesis to its default setting.

SYNTAX

status **acs_reset_directory_structure**

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command resets the project directory tree for Automated Chip Synthesis to its default setting. The command also resets any changes you make to the default directory structure using the **acs_user_dir** array variable.

SEE ALSO

acs_create_directories(2)
acs_get_path(2)
acs_report_directories(2)
create_pass_directories(2)

acs_set_attribute

Sets Automated Chip Synthesis compile attributes on one or more partitions.

SYNTAX

```
status acs_set_attribute  
[-partitions partitions]  
  attribute_name  
  [attribute_value]
```

Data Types

<i>partitions</i>	list or collection
<i>attribute_name</i>	string
<i>attribute_value</i>	data type

ARGUMENTS

-partitions *partitions*

Specifies a partition design list or collection. The command checks each design to determine if it is a partition. If the design is not a partition, a warning message occurs, and the command continues with the attribute setting on that design.

If this option is not specified, the attribute is set on all of the partitions.

attribute_name

Specifies the name of the attribute that you want to set. This argument is required.

attribute_value

Specifies the attribute value that you want to set.

If this option is not specified, the command uses the default value of the attribute, if any. If there is no default value, the attribute value is set to an empty string ("").

DESCRIPTION

The **acs_set_attribute** command sets the Automated Chip Synthesis supported attributes on partitions.

The following lists the ACS supported attributes, their valid values, and their default value.

Attribute Name	Attribute Value	Default

OptimizationPriorities	area/timing/ area_timing/timing_area	timing_area
PreserveBoundaries	true/false	flow specific
CompileVerify	true/false	false
TestReadyCompile	true/false	false
MaxArea	<area estimate in lib units>	unspecified
CanFlatten	true/false	false
FullCompile	none/low/medium/high	medium
IncrementalCompile	none/low/medium/high	medium
BoundaryCompile	none/low/medium/high/top	top
UltraOptimization	true/false	false
AutoUngroup	false/delay/area [numcells]	false
CompileUltra	true/false	false
TargetCompiler	dc_shell/psyn_shell/fpga_shell	dc_shell

Note: Default value of PreserveBoundaries is true for compile and false for compile_ultra.

EXAMPLES

The following example sets the Automated Chip Synthesis **TestReadyCompile** attribute to **true** on the {STACK_TOP, STACK_MEM} partitions and the ALU design, which is not a partition:

```
prompt> acs_set_attribute TestReadyCompile true \
-partition [get_designs {STACK_TOP STACK_MEM ALU}]
```

WARNING: ALU is not a partition

SEE ALSO

acs_report_attribute(2)
report_partitions(2)
set_compile_partitions(2)

acs_submit

Specifies the compile configuration for normal compile partitions.

SYNTAX

```
status acs_submit
  [-command command]
  [-host host]
  [-exec exec]
  [-show]
```

Data Types

```
command  string
host     string
exec     string
```

ARGUMENTS

-command *command*

Specifies the command to use to submit the compile jobs; for example, "bsub ..." or "qsub ...".

ACS will automatically append the following after the user specified command "-o <compile_log>" so the user specified command must support the -o option for specifying a log file. If the command is machine executable which does not support -o option, then the executable can be invoked from a shell script supporting -o option.

If you do not specify this option, ACS uses the default batch submission command, "bsub -K -P".

-host *host*

Specifies from which host to submit the compile jobs.

If you do not specify this option, ACS submits the compile jobs from the your host machine.

-exec *exec*

Specifies which executable file to use for the compile jobs.

If you do not specify this option, ACS uses the executable file specified by the `acs_dc_exec` variable (if you are using a logic synthesis flow) If it is not defined, ACS uses the current executable file.

-show

Prints the current settings of the following variables: `acs_submit_command`, `acs_submit_host`, and `acs_submit_exec`.

DESCRIPTION

The **acs_submit** command specifies the compile configuration for normal compile partitions, including the batch submission command, the executable file, and the host used to run or submit the jobs.

NOTE: This command applies to normal compile partitions only. For information about large compile partitions and setting the compile configuration for them, see the **acs_submit_large** man page.

SEE ALSO

acs_submit_large(2)
set_compile_partitions(2)
report_partitions(2)

acs_submit_large

Specifies the compile configuration for large compile partitions.

SYNTAX

```
status acs_submit_large  
  [-command command]  
  [-host host]  
  [-exec exec]  
  [-partitions part_list]  
  [-show]
```

Data Types

```
command  string  
host     string  
exec     string  
part_list list
```

ARGUMENTS

-command *command*

Specifies the command to use to submit the compile jobs for large compile partitions.

ACS will automatically append the following after the user specified command "-o <compile_log>" so the user specified command must support the -o option for specifying a log file. If the command is machine executable which does not support -o option, then the executable can be invoked from a shell script supporting -o option.

If you do not specify this option, ACS uses the default batch submission command, "bsub -K -P".

-host *host*

Specifies from which host to submit the compile jobs for the large compile partitions.

If you do not specify this option, ACS submits the compile jobs from the your host machine.

-exec *exec*

Specifies which executable file to use for the compile jobs for the large compile partitions.

If you do not specify this option, ACS uses the executable file specified by the `acs_dc_exec` variable (if you are using a logic synthesis flow) If this variables is not defined, ACS uses the current executable file.

-partitions *part_list*

Specifies a list of partitions which are "large partitions". The top-level design is always considered a large partition, whether you explicitly specify it or not.

-show

Prints the current compile configuration for large compile partitions.

DESCRIPTION

The **acs_submit_large** command specifies the compile configuration for large compile partitions, including the batch submission command, the executable file, the host used to run or submit the jobs, and which partitions are considered large partitions.

SEE ALSO

acs_submit(2)
set_compile_partitions(2)
report_partitions(2)

acs_write_html

Creates an HTML page in the destination directory.

SYNTAX

```
status acs_write_html  
-destination pass_name  
[-acs_work_dir acs_working_directory]
```

Data Types

```
pass_name          string  
acs_working_directory string
```

ARGUMENTS

-destination *pass_name*

Specifies the destination pass directory name for this run. If the *pass_name* directory does not exist, the HTML page is not created.

-acs_work_dir *acs_working_directory*

Specifies the ACS working directory. The default is the current directory.

DESCRIPTION

The **acs_write_html** command creates an HTML page in the destination directory. The HTML page contains links to compile scripts, constraint files, and logfiles generated with the **acs_compile_design**, **acs_refine_design**, or **acs_recompile_design** command. The command also provides links to user-created constraint files and script files.

The HTML file is created once during the preparation stage of **acs_compile_design**, **acs_refine_design**, and **acs_recompile_design**. The file is updated again after compiling the top-level partition.

EXAMPLE

The following example creates an HTML page for *top_design_name* in the *pass0* destination directory:

```
prompt> acs_write_html top_design_name -destination pass0
```

SEE ALSO

`acs_recompile_design(2)`

`acs_refine_design(2)`

`acs_merge_design(2)`

add_module

Reads in a specified library file containing module functional information and uses it to update an existing technology library.

SYNTAX

```
status add_module
  [-overwrite]
  [-force]
  [-permanent]
  file_name
  library_name
  [-no_warnings]
```

Data Types

```
file_name      string
library_name   string
```

ARGUMENTS

-overwrite

Indicates that a group declared in *file_name* is to overwrite any group in *library_name* having the same name. Only groups added by **add_module** or **model** can be overwritten. Groups included in the original library and groups added with the **-permanent** option cannot be modified. If **-force** is specified, some library-level groups (wire_load, power_supply, and operating_conditions) included in the original library can be modified. The other groups included in the original library and groups added with **-permanent** cannot be modified.

-force

Indicates that some library-level groups and attributes declared in *file_name* can overwrite the same groups and attributes included in the original library. The library-level groups are as follows:

```
wire_load
power_supply
operating_conditions.
```

The library-level attributes are as follows:

```
<k_factors>
in_place_swap_mode
default_wire_load_mode
default_wire_load_selection
default_wire_load_capacitance
default_wire_load_resistance
```

default_wire_load_area
default_operating_conditions

-permanent

Indicates to store groups declared in *file_name* in the library in such a way that they cannot be overwritten.

file_name

Specifies the name of the file in which one or more new groups containing module functional information are listed. The syntax of this file is the same as that expected by **read_lib**, except that the file contains only library-level groups without the library() {} group definition.

library_name

Specifies the name of the library to update. The library must reside in memory. This option can be a simple filename with no directory specification and, therefore, no slash ("/") (for example, test_db or library.db). Simple filenames must be found in a directory listed in the search_path. Alternatively, this option can be a complex filename, which has a directory specification (for example, ~synopsys/dc/test.lib). Complex filenames are not included in the search_path.

-no_warnings

Indicates that all warning messages are suppressed. The default is to issue all warning messages. Warning messages can identify serious library problems; use this option sparingly.

DESCRIPTION

This command reads in a library file and adds to the specified library only those groups with module functional information declared in the library file. Groups are added to the library as if they were entered at the end of the original text of the library. If the tool finds any errors while processing a group, it does not add that group to the library. You do not need a Library Compiler License to add groups with module functionality to a technology library.

For details on library file syntax, see the Library Compiler manuals.

EXAMPLES

The following example adds the library file *add_ram.lib*, which contains module information, (that is, a RAM group) to the library memory.

```
prompt> add_module add_ram.lib memory
```

The following example adds the library file *romgen.lib*, which contains a cell description of a ROM cell, to the cmos library. The **-permanent** option indicates that the new cell cannot be overwritten.

```
prompt> add_module -permanent romgen.lib cmos
```

SEE ALSO

read_lib(2)

update_lib(2)
write_lib(2)

add_parameter

Define parameters that can be overridden by `apply_power_model`.

SYNTAX

```
status add_parameter
      parameter_name
      [-default default_value]
      [-type type_name]
      [-description description text]
```

Data Types

```
parameter_name  string
default_value   string
type_name       buildtime, runtime, or rate
description     text string
text            description
```

ARGUMENTS

parameter_name

Specifies the name of the parameter to be defined inside a power model.

-default *default_value*

Specifies the default value of the parameter if there is no parameter mapping specified in the `apply_power_model` command.

-type *type_name*

When -type option is specified, the parameter defined is used for system-level IP power model. The parameter scope is within the power model only and power models and power functions cannot access parameters that are defined outside of the power model in which they are used.

Three types of parameters can be defined as follows:

1. Build time - for parameters that remain unchanged during run time
2. Run time - for parameters that can change during run time
3. Rate - for parameters that represent rate-based quantities that can change during run time

-description *description text*

Text description of the parameter.

DESCRIPTION

The `add_parameter` command is used to define parameters for use within a power model. A parameter can be used just like a regular Tcl variable by prepending the parameter name with the `$` keyword.

This command can only be defined within the `define_power_model` command and applied to a macro instance through the `apply_power_model` command. It is an error if this command is executed through the `load_upf` command.

The `-parameters` option of the `apply_power_model` command can change the value of a parameter during application of the power model on a macro instance. If the parameter is not found in the `-parameter` option, default value specified in `add_parameter` will be used.

A power model provides a self-contained environment for variables. Parameters defined outside a power model are not visible inside the power model. Parameters defined within a power model are only visible inside the power model.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
define_power_model MODEL -for {lib_cell} {  
  add_parameter PD_NAME -default "PD" -description "power domain in model"  
  add_parameter process -type buildtime -default 1.0  
  add_parameter cache_miss -type rate -default 0.02  
  create_supply_net VDD  
  create_supply_net VSS  
  create_supply_set SS -function {power VDD} -function {ground VSS}  
  create_power_domain $PD_NAME -supply {primary SS} -include_scope  
}  
  
apply_power_model MODEL -parameters {{PD_NAME pd_macro}}
```

SEE ALSO

`define_power_model(2)`
`apply_power_model(2)`
`report_power_model(2)`

add_pg_pin_to_db

Converts a non-PG-pin-based logic library in compiled format (.db file) into a PG-pin-based logic library in .db format.

SYNTAX

```
status add_pg_pin_to_db  
  input_db_filename  
  -output pg_db_filename  
  [-mw_library_name mw_lib_name]  
  [-pg_map_file map_file]  
  [-pg_map_template template_filename]  
  [-expanded]  
  [-fast]  
  [-force_update]  
  [-verbose]
```

Data Types

```
input_db_filename  string  
pg_db_filename    string  
mw_lib_name       list  
map_file          string  
template_filename string
```

ARGUMENTS

input_db_filename

Specifies the name of the existing logic library in compiled format (.db file) that lacks PG pin data.

-output *pg_db_filename*

Specifies the name of the new PG-pin-based .db file generated by the conversion. This name must be different from the name of the input .db file unless the **-force_update** option is used.

-mw_library_name *mw_lib_name*

Specifies the names of one or more Milkyway libraries containing the FRAM views of the input .db library file. The FRAM views provide PG pin information used to update the library.

-pg_map_file *map_file*

Specifies the name of a text file that defines the mapping between non-PG-pin-based data and PG-pin-based data.

-pg_map_template *template_filename*

Specifies the file name a map template. If you use this option, the command generates only a map template file; it does not perform a library update. You can edit the generated template file and use it to specify the PG mapping for a subsequent **add_pg_pin_to_db** command.

-expanded

Generates an expanded rather than compressed PG map template. By default, when you use the **-pg_map_template** option, the command generates a compressed PG map template, using wildcards to specify multiple mappings where possible, to reduce the file size. Use this option to generate a PG map template with the wildcards expanded to individual names.

-fast

Generates a PG pin library from a non-PG pin library without using a PG map file. This option uses the information available in the Milkyway FRAM view, the input non-PG pin library, or a combination of both. It is recommended that you use the FRAM view, which helps identify the names of the cell's PG pins. If you omit the **-mw_library_name** option, the FRAM view is not used and the **-fast** option can only infer the PG pin names if they are specified in the logic library's Liberty (.lib file) attributes: **rail_connection**, **input_signal_level**, and **output_signal_level**.

-force_update

Forces the command to update the library data based on the input data and overwrite the existing data in the library. By default, the command writes out a new .db library and retains the existing library.

-verbose

Writes out all sections of the PG map data, whether generated automatically or expanded from a compressed PG map file.

DESCRIPTION

The **add_pg_pin_to_db** command converts a non-PG-pin-based logic library in compiled format (.db file) into a PG-pin-based logic library and writes out the new, updated library in .db format.

The command converts and updates a logic library automatically from the old **rail_connection** Liberty syntax or from a format that is not based on PG pin syntax to a library with PG pin syntax. You specify an existing logic library in .db format and the related physical libraries with Milkyway FRAM views, if available, and a PG map file.

The command gets information it needs to perform the conversion from the FRAM views in the Milkyway library or from the PG map file, or both. The PG map file is a text file that specifies the mapping between signal pins and their related power and ground pins, between PG pins and their corresponding voltage names, and between voltage names and voltage values.

You can use the **add_pg_pin_to_db** command with the **-pg_map_template** option to automatically generate a PG map template file, which you can then edit to provide the specific mapping information. When you use this option, the command only writes out the PG map template file and does not actually perform the library conversion. You first need to edit the template file, then use the command again with the **-pg_map_file** option to perform the library conversion.

The options you can use depend on whether a Milkyway library with FRAM views is available:

- With a complete Milkyway library

If all cells have single PG rail (one power rail and one ground rail), the **-pg_map_file** option is not required; use the **-mw_library_name** option to allow the command to derive the mapping from the Milkyway FRAM view. If some cells have multiple PG rails, the **-pg_map_file** option is required.

- Without a Milkyway library (.db library only)

If you specify only a logic library (.db file), without using the **-mw_library_name** option, you need to use the **-pg_map_file** option, whether the library uses single or multiple PG rails.

- With a partial Milkyway library

If some cells exist in both the logic library and the Milkyway library while others exist only in the logic library, you need to use the **-pg_map_file** option for the cells that are missing from the Milkyway library.

For more information, see "Adding PG Pin Syntax to Logic Libraries" in the "UPF Library Preparation Application Notes," available on SolvNet.

EXAMPLES

The following example writes out a PG map template file named "pg.map" using information from the compiled logic library "old.db" and the corresponding FRAM views in the Milkyway library named "mw_lib." This command does not convert the library; it only writes out the PG map template file, which you can edit and use later for library conversion.

```
prompt> add_pg_pin_to_db old.db -mw_library_name {mw_lib} \  
-pg_map_template pg.map
```

The following example converts a non-PG-pin-based logic library named "old.db" into a PG-pin-based library named "pg.db."

```
prompt> add_pg_pin_to_db old.db -mw_library_name {mw_lib} \  
-pg_map_file pg.map -output pg.db
```

SEE ALSO

add_pg_pin_to_lib(2)

add_pg_pin_to_lib

Converts a non-PG-pin based logic library in Liberty format (.lib file) into a PG-pin-based logic library in .lib format.

SYNTAX

```
status add_pg_pin_to_lib
  input_lib_filename
  -output pg_lib_filename
  [-mw_library_name mw_lib_name]
  [-pg_map_file map_file]
  [-pg_map_template template_filename]
  [-common_shell_path common_shell_path]
  [-expanded]
  [-fast]
  [-force_update]
  [-verbose]
```

Data Types

```
input_lib_filename  string
pg_lib_filename    string
mw_lib_name        list
map_file           string
template_filename  string
common_shell_path  string
```

ARGUMENTS

input_lib_filename

Specifies the name of the existing logic library in Liberty format (.lib file) that lacks PG pin data.

-output *pg_lib_filename*

Specifies the name of the new PG-pin-based .lib file generated by the conversion. This name must be different from the name of the input .lib file unless the **-force_update** option is used.

-mw_library_name *mw_lib_name*

Specifies the names of one or more Milkyway libraries containing the FRAM views of the input .lib library file. The FRAM views provide PG pin information used to update the library.

-pg_map_file *map_file*

Specifies the name of a text file that defines the mapping between non-PG-pin-based data and PG-pin-based data, including the

power management attributes.

-pg_map_template *template_filename*

Specifies the file name a map template. If you use this option, the command generates only a map template file; it does not perform a library update. You can edit the generated template file and use it to specify the PG mapping for a subsequent **add_pg_pin_to_lib** command.

-common_shell_path *common_shell_path*

Specifies a common shell path where **lc_shell** or **dc_shell** resides, to compile the input .lib using an older tool version. By default, the current Library Compiler or Design Compiler tool version is used. Use this option for older .lib files, when the latest utility version cannot compile the input .lib file due to new checking applied in the current version.

-expanded

Generates an expanded rather than compressed PG map template. By default, when you use the **-pg_map_template** option, the command generates a compressed PG map template, using wildcards to specify multiple mappings where possible, to reduce the file size. Use this option to generate a PG map template with the wildcards expanded to individual names.

-fast

Generates a PG pin library from a non-PG pin library without using a PG map file. This option uses the information available in the Milkyway FRAM view, the input non-PG pin library, or a combination of both. It is recommended that you use the FRAM view, which helps identify the names of the cell's PG pins. If you omit the **-mw_library_name** option, the FRAM view is not used and the **-fast** option can only infer the PG pin names if they are specified in the logic library's Liberty (.lib file) attributes: **rail_connection**, **input_signal_level**, and **output_signal_level**.

-force_update

Forces the command to update the library data based on the input data and overwrite the existing data in the library. By default, the command writes out a new .lib library and retains the existing library.

-verbose

Writes out all sections of the PG map data, whether generated automatically or expanded from a compressed PG map file.

DESCRIPTION

The **add_pg_pin_to_lib** command converts a non-PG-pin based logic library in Liberty format (.lib file) into a PG-pin-based logic library and writes out the new, updated library in .lib format.

The command converts and updates a logic library automatically from the old **rail_connection** Liberty syntax or from a format that is not based on PG pin syntax to a library with PG pin syntax. You specify an existing logic library in .lib format and the related physical libraries with Milkyway FRAM views, if available, and a PG map file.

The command gets information it needs to perform the conversion from the FRAM views in the Milkyway library or from the PG map file, or both. The PG map file is a text file that specifies the mapping between signal pins and their related power and ground pins, between PG pins and their corresponding voltage names, and between voltage names and voltage values.

You can use the **add_pg_pin_to_lib** command with the **-pg_map_template** option to automatically generate a PG map template file, which you can then edit to provide the specific mapping information. When you use this option, the command only writes out the PG map template file and does not actually perform the library conversion. You first need to edit the template file, then use the command again with the **-pg_map_file** option to perform the library conversion.

The options you can use depend on whether a Milkyway library with FRAM views is available:

- With a complete Milkyway library

If all cells have single PG rail (one power rail and one ground rail), the **-pg_map_file** option is not required; use the **-mw_library_name** option to allow the command to derive the mapping from the Milkyway FRAM view. If some cells have multiple PG rails, the **-pg_map_file** option is required.

- Without a Milkyway library (.lib library only)

If you specify only a logic library (.lib file), without using the **-mw_library_name** option, you need to use the **-pg_map_file** option, whether the library uses single or multiple PG rails.

- With a partial Milkyway library

If some cells exist in both the logic library and the Milkyway library while others exist only in the logic library, you need to use the **-pg_map_file** option for the cells that are missing from the Milkyway library.

For more information, see "Adding PG Pin Syntax to Logic Libraries" in the "UPF Library Preparation Application Notes," available on SolvNet.

EXAMPLES

The following example writes out a PG map template file named "pg.map" using information from the Liberty logic library "old.lib" and the corresponding FRAM views in the Milkyway library named "mw_lib." This command does not convert the library; it only writes out the PG map template file, which you can edit and use later for library conversion.

```
prompt> add_pg_pin_to_lib old.lib -mw_library_name {mw_lib} \  
-pg_map_template pg.map
```

The following example converts a non-PG-pin-based logic library named "old.lib" into a PG-pin-based library named "pg.lib."

```
prompt> add_pg_pin_to_lib old.lib -mw_library_name {mw_lib} \  
-pg_map_file pg.map -output pg.lib
```

SEE ALSO

add_pg_pin_to_db(2)

add_port_state

Adds state information to a supply port.

SYNTAX

```
string add_port_state  
  supply_port_name  
  -state {state_name state_value}
```

Data Types

```
supply_port_name  string  
state_name        string  
state_value       string
```

ARGUMENTS

supply_port_name

Specifies the name of the supply port. Hierarchical names are allowed.

-state {*state_name state_value*}

Specifies the name and value of a state of the supply port. You can repeat this option. The state value can be one of the following:

- A single floating point number that represents the nominal voltage for the specified state. For example, to define a state called s88 with a nominal voltage of 0.88, use the following syntax:

-state {s88 0.88}
- Three floating point numbers that represent the minimum, nominal, and maximum voltages of the specified state, respectively. For example, to define a state called active_state with a minimum voltage of 0.88, a nominal voltage of 0.90, and a maximum voltage of 0.92, use the following syntax:

```
-state {active_state 0.88 0.90 0.92}
```

NOTE: It is an error if minimum voltage is greater than (>) the nominal voltage or the nominal voltage is greater than (>) the maximum voltage.

- The keyword **off**, which indicates an inactive state. For example, to define an inactive state called off_state, use the following syntax:

```
-state {off_state off}
```

DESCRIPTION

This command adds state information to a supply port. The first occurrence of the *supply_port_name* option is the default state of the supply port. You can use this to represent off-chip supply sources that are not driven by the testbench. This option defines the voltage level supplied to the chip. It provides a convenient shortcut to facilitate verification and analysis without requiring the creation of a power domain and a supply network within the verification environment. An error occurs if *supply_port_name* does not already exist before executing this command.

This command returns the fully-qualified name from the current scope of the created port or a null string if the port is not created.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the states and voltages of a supply port named VN1:

```
prompt> add_port_state VN1 \  
-state {active_state 0.88 0.90 0.92} \  
-state {off_state off}
```

SEE ALSO

add_pst_state(2)
create_pst(2)
report_pst(2)

add_power_state

Adds state information to a supply set or a group or a domain.

SYNTAX

```
status add_power_state
[-supply | -group | -domain] object_name
[-simstate {simstate}]
[-update]
[-state state_name {[-supply_expr {supply_expression}]
                    [-logic_expr {logic_expression}]
                    [-simstate {simstate}]
                    [-illegal]]}]*
```

Data Types

```
object_name    string
simstate       string
state_name     string
supply_expression string
logic_expression string
```

ARGUMENTS

-supply | -group | -domain *object_name*

Specifies the name of a supply set, or the name of a group created with the **create_power_state_group** command, or the name of a domain. The name should be a simple (nonhierarchical) name.

-state *state_name*

Specifies the name of the state of the supply set or the group or the domain. The name should be a simple (nonhierarchical) name. This option can be specified multiple times in the same command.

-supply_expr {*supply_expression*}

Specifies a Boolean expression in terms of nets and their netstates. This option can only be specified when the *object_name* is a supply set. The only operator allowed in the Boolean expression is && (AND operator). Each subexpression in the Boolean expression specifies the net and netstate definitions. Each subexpression must have the following syntax:

```
net == netstate
```

The *net* can be power or ground and the *netstate* syntax must be one of the following:

```
status
```

```
`{status}
`{status, nom}
`{status, min, max}
`{status, min, nom, max}
{status}
{status nom}
{status min max}
{status min nom max}
```

- *status* can be OFF or FULL_ON
- *min*, *nom*, and *max* are floating-point numbers representing the minimum, nominal, and maximum voltages of the specified state, respectively.

NOTE: It is an error if *min* is greater than (>) *nom* or if *nom* is greater than (>) *max*.

- If the *status* is FULL_ON, at least one voltage must be defined.

-logic_expr {logic_expression}

If the *object_name* is a supply set, then this option specifies a SystemVerilog Boolean expression in terms of logic nets and supply nets. The **-logic_expr** option does not affect implementation and is intended to be used by simulation tools, to read and write transparently.

If the *object_name* is a group or a domain, then this option specifies a Boolean expression in terms of supply sets/groups/domains/PSTs and their states. The only operator allowed in the Boolean expression is &&.

-simstate {simstate}

Specifies a simstate for the power states associated with a supply set. Valid values are NORMAL, CORRUPT_ON_ACTIVITY, CORRUPT, NOT_NORMAL, CORRUPT_STATE_ON_CHANGE, CORRUPT_STATE_ON_ACTIVITY and CORRUPT_ON_CHANGE. This option can only be specified when the *object_name* is a supply set. The **-simstate** option does not affect implementation and is intended to be used by simulation, so the tool reads and writes the information specified with this option transparently.

This option can be specified for every state that is defined with the **add_power_state** command:

```
add_power_state -state state_name {-simstate simstate_value}.
```

This option can also be specified globally for the entire **add_power_state** command:

```
add_power_state -simstate simstate_value -state state_name {...}
```

However, the UPF standard does not use the **-simstate** globally. It is retained only for backward compatibility purposes and will be obsolete in a future release.

-update

Specifies additional power states (supply set states or group states or domain states) to add to an object. Also specifies additional power state intent to add to an existing state.

For supply sets, this option can be used to add incremental information (**-logic_expr**, **-supply_expr**, **-simstate**) to an existing supply set state.

For groups and domains, this option can be used to add incremental information (**-logic_expr**) to an existing group or domain state.

-illegal

Indicates that the state defined (for a supply set or a group) is an illegal state.

DESCRIPTION

This command adds state information to a supply set or a group.

If the object specified is a supply set, then use this command to set power states on the nets of a supply set and then use them as supplies for the power state table. If the supply set does not already exist, this command issues an error.

If the object specified is a group, then use this command to create the power state table by using supply sets, other groups or PSTs in the **-logic_expr** option. The group should have been previously created using the **create_power_state_group** command. If the group does not already exist, this command issues an error. To add additional group states to a group, the **-update** option must be used.

EXAMPLES

The following example shows the states and voltages of a supply set, SS1:

```
prompt> add_power_state SS1 \
      -state HVp {-supply_expr {power == `{FULL_ON, 0.88, 0.90, 0.92}}}
```

```
prompt> add_power_state SS1 \
      -state HPg {-supply_expr {ground == `{OFF}}}
```

The following example defines LV state, but omitting the details on voltages for SS2 supply set:

```
prompt> add_power_state SS2 \
      -state LV {}
```

The following example defines a state ON for supply set SS1 where the supply_expression is a Boolean expression of power and ground.

```
prompt> add_power_state -supply SS1\
      -state ON {-supply_expr {power == {FULL_ON 0.8} && ground == {FULL_ON 0}}}
```

The following example shows how power states can be added to a group. Here 'GROUP1' is a power state group that has previously been created with the **create_power_state_group** command. GS1 and GS2 are the names of the group states created. The system is in state GS1 when supply set SS1 is in state ON1, and group GROUP2 defined in scope MID1 is in state ON1, and the PST PST1 defined in scope MID2 is in state ON1.

```
prompt> add_power_state -group GROUP1 \
      -state GS1 {-logic_expr {SS1 == ON1 && MID1/GROUP2 == ON1 && MID2/PST1 == ON1}}
      -state GS2 {-logic_expr {SS1 == ON2 && MID1/GROUP2 == ON2 && MID2/PST1 == ON2}}
```

SEE ALSO

create_power_state_group(2)
 add_pst_state(2)
 create_pst(2)

```
create_supply_set(2)  
report_pst(2)
```

add_pst_state

Defines the states of each of the supply nets for one possible state of the design.

SYNTAX

```
status add_pst_state  
    state_name  
    -pst table_name  
    -state supply_states
```

Data Types

<i>state_name</i>	string
<i>table_name</i>	string
<i>supply_states</i>	list

ARGUMENTS

state_name

Specifies the name of the power state.

-pst *table_name*

Specifies the power state table (PST) to which this state applies.

-state *supply_states*

Lists the supply net state names in the corresponding order of the **-supplies** option listing in the **create_pst** command.

DESCRIPTION

The **add_pst_state** command defines the states of each of the supply nets for one possible state of the design. It is an error if the number of supply state names is different from the number of supply nets within the power state table.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines the supply net state names for the s1, s2 and s3 power states:

```
create_pst pt -supplies { PN1 PN2 SOC/OTC/PN3 }  
add_pst_state s1 -pst pt -state { s08 s08 s08 }  
add_pst_state s2 -pst pt -state { s08 s08 off }  
add_pst_state s3 -pst pt -state { s08 s09 off }
```

SEE ALSO

add_port_state(2)
create_pst(2)
report_pst(2)

add_row

Creates a list of rows in the design.

SYNTAX

```
status add_row
  -within {coordinates}
  [-direction horizontal | vertical]
  [-flip_first_row]
  [-no_double_back]
  [-minimal_channel_height channel_height]
  [-no_start_from_first_row]
  [-tile_name tile_name]
  [-snap_to_row_direction {wire_track | tile_width | none}]
  [-snap_to_orthogonal_row_direction {existing_row | wire_track | none}]
  [-left_offset left_offset]
  [-right_offset right_offset]
  [-top_offset top_offset]
  [-bottom_offset bottom_offset]
  [-allow_overlap]
```

Data Types

<i>coordinates</i>	list
<i>channel_height</i>	float
<i>tile_name</i>	string
<i>left_offset</i>	float
<i>right_offset</i>	float
<i>top_offset</i>	float
<i>bottom_offset</i>	float

ARGUMENTS

-within {*coordinates*}

Specifies the boundary within which to create rows. For a rectangular region, specify the coordinates for the lower-left and upper-right corners as {ll_x ll_y} {ur_x ur_y}. For a rectilinear region, specify a list of rectilinear coordinates in either clockwise or counterclockwise order.

-direction horizontal | vertical

Specifies the direction of rows to be created. By default, the direction of rows is the same as the direction of the base array where the rows are added.

-flip_first_row

Flips the bottom row in a horizontal core area or the leftmost row in a vertical core area. The **-flip_first_row** and **-no_double_back** options are mutually exclusive. By default, the first row is not flipped.

-no_double_back

Creates rows without flipping one row in each pair. The **-no_double_back** and **-flip_first_row** options are mutually exclusive. By default, the area can contain pairs of rows that are doubled back.

-minimal_channel_height *channel_height*

Specifies the amount of channel height to allocate for routing between rows. By default, the minimum channel height is 0.0.

-no_start_from_first_row

Specifies that the bottom row in a horizontal core area or the leftmost row in a vertical core area is not paired with the second row. By default, the first row can form a row pair with the second row.

-tile_name *tile_name*

Specifies the name of the unit tile to use in rows. By default, the tile name is **unit**.

-snap_to_row_direction {*wire_track* | *tile_width* | *none*}

Specifies the snapping type in the row direction.

If you specify a snapping type of **wire_track** for a design with horizontal rows, the tool aligns the vertical-layer wire tracks of the tile and the design. For a design with vertical rows, the tool aligns the horizontal-layer wire tracks of the tile and the design.

If you specify a snapping type of **tile_width** for a design with horizontal rows, the tool separates the left edge of the base array boundary and the starting left edge of the row by using a multiple of the tile width. For a design with vertical rows, the tool separates the bottom edge of the base array boundary and the bottom edge of the first row by using a multiple of the tile width.

If you specify a snapping type of **none** for a design with horizontal rows, the tool creates each new row from the left edge as specified by the **-within** option. For a design with vertical rows, the tool creates each new row from the bottom edge as specified by the **-within** option.

By default, the snapping type is **wire_track**.

-snap_to_orthogonal_row_direction {*existing_row* | *wire_track* | *none*}

Specifies the snapping type in the orthogonal row direction.

If you specify an orthogonal snapping type of **existing_row**, the tool attempts to align the new row with an existing row that is within one tile height of the specified location. The existing row must have the same tile pattern and orientation as the first new row. If the tool cannot find an existing row that meets the criteria, the tool snaps the new row to a wire track.

If you specify a snapping type of **wire_track** for a design with horizontal rows, the tool aligns the horizontal-layer wire track of the tile and the design. For a design with vertical rows, the tool aligns the vertical-layer wire tracks of the tile and the design.

If you specify a snapping type of **none** for a design with horizontal rows, the tool creates the first row at the bottom edge as specified by the **-within** option. For a design with vertical rows, the tool creates the first row at the left edge as specified by the **-within** option.

By default, the orthogonal row snap type is **existing_row**.

-left_offset *left_offset*

Specifies the minimum space between the new rows and the left boundary specified by the **-within** option. By default, the left offset is 0.

-right_offset *right_offset*

Specifies the minimum space between the new rows and the right boundary specified by the **-within** option. By default, the right offset is 0.

-top_offset *top_offset*

Specifies the minimum space between the new rows and the top boundary specified by the **-within** option. By default, the top offset is 0.

-bottom_offset *bottom_offset*

Specifies the minimum space between the new rows and the bottom boundary specified by the **-within** option. By default, the bottom offset is 0.

-allow_overlap

Specifies that overlapped rows are allowed. By default, overlapped rows are not allowed, whether they are same height or not.

DESCRIPTION

This command adds rows to the specified area.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates rows in a rectangular region.

```
prompt> add_row -within {{100 100} {300 300}} -tile_name unit1
info: created 6 rows.
1
```

The following example creates rows in the specified rectilinear region.

```
prompt> add_row -within {{30.0 30.0} {30.0 50.0} \
    {40.0 50.0} {40.0 60.0} {50.0 60.0} {50.0 40.0} \
    {60.0 40.0} {60.0 30.0} {30.0 30.0}} -tile_name unit2
```

```
Info: 18 rows are added.
1
```

The following example creates rows and sets left/right offset of 5 units and top/bottom offset of 10 units.

```
prompt> add_row -within {{100 100} {300 300}} \
    -left_offset 5 -right_offset 5 \
    -top_offset 10 -bottom_offset 10 \
```

```
Info: 6 rows are added.
1
```

The following example creates rows with a channel height of 3.69 units between the rows.

```
prompt> add_row -within {{30 148} {30 159} \  
    {645 159} {645 166} {817 166} \  
    {817 159} {1432 159} {1432 148}} \  
    -minimal_channel_height 3.69  
Info: created 4 rows.  
1
```

The following example creates rows that overlap with the existing rows in the region.

```
prompt> add_row -within {{30 148} {30 159} \  
    {645 159} {645 166} {817 166} \  
    {817 159} {1432 159} {1432 148}} \  
    -allow_overlap  
Warning: Existing row (id = 11279) found in the specified new row creation region. (MWUI-183)  
Warning: Existing row (id = 11280) found in the specified new row creation region. (MWUI-183)  
Warning: Existing row (id = 11281) found in the specified new row creation region. (MWUI-183)  
Warning: Existing row (id = 11282) found in the specified new row creation region. (MWUI-183)  
Warning: Existing row (id = 11283) found in the specified new row creation region. (MWUI-183)  
Info: created 5 rows.  
1
```

SEE ALSO

cut_row(2)
write_floorplan(2)

add_state_transition

SYNTAX

```
status add_state_transition
  [-supply | -domain | -group] object_name
  [-update]
  [-transition transition_name {through_list}]]]*
  [-complete]
```

Data Types

transition_name string

ARGUMENTS

-supply | -domain | -group *object_name*

Optional Argument. Specifies the name of a supply set, power domain or power state group that the transitions will be added to. If no type is specified, the specified object can be any of the following types:

- Power Domain - Supply Set - Power State Group - Power State Table - Supply Port - Supply Net - Power Switch

-update

Optional Argument. Used when updating an already existing transition.

-transition *transition_name* {

Specifies the name of a transition to be added to the object. This option can be specified multiple times in the same command.

-complete

Any new transitions defined after the command with -complete is given will result in an error.

DESCRIPTION

This command defines named state transitions between power states of an object. This is a UPF 3.0 command and it replaces the describe_state_transition command. If the object specified doesn't exist or isn't any of the specified types, the command issues an error.

Multiple transitions can be defined in the same command, and the initial and final states of each transition can be specified either using two separate **-from** and **-to** lists or a **-paired** list. If none of these are specified the command issues an error. Furthermore, the

same state can't be an initial and final state of the same transition, and specifying this will cause the command to return an error.

Synopsys has a specific extension consisted of the **-through** option, which is a list of intermediate states to the **-from** and **-to** lists. If this option is specified without the **-from** or **-to** option, the command issues an error.

Design Compiler doesn't preserve this command through hierarchical flow, a warning will be issued from characterize when this command is seen.

Design Compiler will only parse and preserve this command.

EXAMPLES

Following examples show different usages of **add_state_transition** command and the outcome.

```
prompt> add_state_transition -supply MID_SS -transition {turn_on -from [list OFF PWR_OFF] -to [list ON PWR_ON] -legal} -tra
1
```

```
prompt> add_state_transition -domain supply_set_1 -transition {failed_transition -from OFF -to ON -legal}
Error: Object supply_set_1 isn't a defined power domain. (UPF-901)
```

```
prompt> add_state_transition MID_PD -transition {failed_transition -from ON -through TRAN}
Error: State list -through can't be defined if -from or -to lists are empty. (UPF-897)
```

```
prompt> add_state_transition MID_SS -transition {failed_transition}
Error: State lists -from and -to can't be empty if -paired state list is not defined. (UPF-896)
```

```
prompt> add_state_transition MID3_SS -transition {failed_transition -from OFF -to ON}
Error: Object MID3_SS wasn't found (UPF-899)
```

SEE ALSO

`describe_state_transition(2)`

add_to_collection

Adds objects to a collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

```
collection add_to_collection
  [-unique]
  collection1
  object_spec
```

Data Types

```
collection1  collection
object_spec list
```

ARGUMENTS

-unique

Indicates that duplicate objects are to be removed from the resulting collection. By default, duplicate objects are not removed.

collection1

Specifies the base collection to which objects are to be added. This collection is copied to the result collection, and objects matching *object_spec* are added to the result collection. The *collection1* option can be the empty collection (empty string), subject to some constraints, as explained in the DESCRIPTION section.

object_spec

Specifies a list of named objects or collections to add.

If the base collection is heterogeneous, only collections can be added to it.

If the base collection is homogeneous, the object class of each element in this list must be the same as in the base collection. If it is not the same class, it is ignored. From heterogeneous collections in the *object_spec*, only objects of the same class of the base collection are added. If the name matches an existing collection, the collection is used. Otherwise, the objects are searched for in the database using the object class of the base collection.

The *object_spec* has some special rules when the base collection is empty, as explained in the DESCRIPTION section.

DESCRIPTION

The **add_to_collection** command allows you to add elements to a collection. The result is a new collection representing the objects in the *object_spec* added to the objects in the base collection.

Elements that exist in both the base collection and the *object_spec*, are duplicated in the resulting collection. Duplicates are not removed unless you use the *-unique* option. If the *object_spec* is empty, the result is a copy of the base collection.

If the base collection is homogeneous, the command searches in the database for any elements of the *object_spec* that are not collections, using the object class of the base collection. If the base collection is heterogeneous, all implicit elements of the *object_spec* are ignored.

When the *collection1* argument is the empty collection, some special rules apply to the *object_spec*. If the *object_spec* is non-empty, there must be at least one homogeneous collection somewhere in the *object_spec* list (its position in the list does not matter). The first homogeneous collection in the *object_spec* list becomes the base collection and sets the object class for the function. The examples show the different errors and warnings that can be generated.

The **append_to_collection** command has similar semantics as the **add_to_collection** command; however, the **append_to_collection** command can be much more efficient in some cases. For more information about the command, see the man page.

For background on collections and querying of objects, see the **collections** man page.

EXAMPLES

The following example from PrimeTime uses the **get_ports** command to get all of the ports beginning with 'mode' and then adds the "CLOCK" port.

```
prompt> set xports [get_ports mode*]
{mode[0] mode[1] mode[2]}
prompt> add_to_collection $xports [get_ports CLOCK]
{mode[0] mode[1] mode[2] CLOCK}
```

The following example from PrimeTime adds the cell u1 to a collection containing the SCANOUT port.

```
prompt> set so [get_ports SCANOUT]
{SCANOUT}
prompt> set u1 [get_cells u1]
{u1}
prompt> set het [add_to_collection $so $u1]
{u1}
prompt> query_objects -verbose $het
{port:SCANOUT cell:u1}
```

The following examples show how the **add_to_collection** command behaves when the base collection is empty. Adding two empty collections yields the empty collection. Adding an implicit list of only strings or heterogeneous collections to the empty collection generates an error message, because no homogeneous collections are present in the *object_spec* list. Finally, as long as one homogeneous collection is present in the *object_spec* list, the command succeeds, even though a warning message is generated. The example uses the variable settings from the previous example.

```
prompt> sizeof_collection [add_to_collection "" ""]
0

prompt> set A [add_to_collection "" [list a $het c]]
Error: At least one homogeneous collection required for argument 'object_spec'
to add_to_collection when the 'collection' argument is empty (SEL-014)
```

```
prompt> add_to_collection "" [list a $het $sp]]
Warning: Ignored all implicit elements in argument 'object_spec'
       to add_to_collection because the class of the base collection
       could not be determined (SEL-015)
{SCANOUT u1 SCANOUT}
```

SEE ALSO

append_to_collection(2)
collections(2)
query_objects(2)
remove_from_collection(2)
sizeof_collection(2)

add_to_rp_group

Adds a cell, hierarchical group, or keepout to an existing relative placement group.

SYNTAX for Leaf Cells

```
status add_to_rp_group
  rp_groups
  -leaf cell_names
  [-column integer]
  [-row integer]
  [-pin_align_name pin_name]
  [-orientation direction]
  [-alignment bottom-left | bottom-right]
  [-num_rows integer]
  [-num_columns integer]
  [-free_placement]
```

Data Types

<i>rp_groups</i>	list or collection
<i>cell_names</i>	cells
<i>pin_name</i>	string
<i>direction</i>	list of strings

SYNTAX for Hierarchical Groups

```
status add_to_rp_group
  rp_groups
  -hierarchy group_name
  [-instance instance_name]
  [-column integer]
  [-row integer]
  [-orientation direction]
  [-alignment bottom-left | bottom-right]
```

Data Types

<i>group_name</i>	list or collection
<i>instance_name</i>	cell

SYNTAX for Keepouts

```
status add_to_rp_group
  rp_groups
  -keepout keepout_name
  [-column integer]
  [-row integer]
  [-width integer]
  [-height integer]
  [-type hard | soft | space]
```

Data Types

keepout_name string

ARGUMENTS

rp_groups

Specifies the relative placement groups in which to add an item. The groups must all be in the same design.

-leaf *cell_names*

Specifies the name of a cell to add to the relative placement groups in *rp_groups*. The specified cell must be in the design that contains the relative placement groups in *rp_groups*. Multiple cells are accepted when the **-free_placement** option is specified.

-column *integer*

Specifies the column position in which to add the item. If you do not specify the column position, the default is zero.

-row *integer*

Specifies the row position in which to add the item. If you do not specify the row position, the default is zero.

-pin_align_name *pin_name*

Specifies the name of the pin to use for pin alignment of this cell with other cells in a group. This overrides the default pin name specified for the relative placement group into which it is being added. This option can only be used when adding a leaf cell with the **-leaf** option.

-orientation *direction*

Specifies the placement orientation of the cell or relative placement group being added. The option functions differently for cells from for relative placement groups. For cells, specify the orientation with respect to the group to which the cell is being added. You can specify a list of possible orientations, and the tool chooses the first legal orientation for the cell. For relative placement groups, the orientation you specify overrides any group orientation that was specified.

You can specify the direction using DEF syntax:

```
DEF syntax:
  N, W, S, E,  FN,  FW,  FS,  FE
```

This option can only be used to add a leaf cell with the **-leaf** option or hierarchical instance with the **-hierarchy** and **-instance** options.

-alignment bottom-left | bottom-right

Specifies the alignment to use when placing the cell or group with respect to its parent group.

If a relative placement group has the **-compress** option set, and you add an element in that group with the **-alignment bottom-right** option, then the alignment type of that element is ignored.

-num_rows integer

Specifies the number of rows to which you add the leaf cell. You can specify multiple rows, ranging from 1 to 255. The default is 1 if you do not specify this option.

-num_columns integer

Specifies the number of columns to which you add the leaf cell. You can specify multiple columns, ranging from 1 to 255. The default is 1 if you do not specify this option. You cannot use the bottom-right and bottom-pin alignment settings for a leaf cell that occupies multiple columns.

-free_placement

Specifies that the cells being provided using **-leaf** option must be placed randomly, that is, similar to the cells of a bound. These cells can be added in multiple locations using the **-num_rows** and **-num_columns** options. The width and height of the placement area is automatically calculated for free placement cells placed inside relative placement group. You might need to specify the **-width** option when free placement cells are being added to boundary columns and the **-height** option when cells are being added to boundary rows. The cell being added should not belong to any other bound.

-hierarchy group_name

Specifies the relative placement group to be added hierarchically to the relative placement groups in *rp_groups*.

-instance instance_name

Specifies the hierarchical cell in which to instantiate the relative placement group specified with the **-hierarchy** option. The cell must be an instance of the reference design that contains the relative placement group specified with **-hierarchy** and must be in the design containing the relative placement groups specified in *rp_groups*. This option can be used only when adding a relative placement group with the **-hierarchy** option.

-keepout keepout_name

Specifies the name of the keepout to be added to the relative placement groups in *rp_groups*. Although the keepout is not a design object, you name the keepout to refer to it after it is created.

-width integer

Specifies the width of the keepout being added when used with **-keepout**. Unit for keepout width is the width of the site row for a particular library. If you do not specify the width, the keepout defaults to the width of the column into which the keepout is being added.

It also specifies the width of the placement area for multiple cells being added with the **-free_placement** option. The unit is the width of one site of site row.

-height integer

Specifies the height of the keepout being added when used with the **-keepout** option. Unit for keepout height is the height of the site row for a particular library. If you do not specify the height, the default value is height of the row of relative placement group to which the keepout is added.

It also specifies the height of the placement area for multiple cells being added with the **-free_placement** option. The unit is the

height of the site row.

-type hard | soft | space

Specifies whether a keepout is **hard**, **soft**, or **space**.

- A **hard** keepout keeps everything out of a location during legalization. This is the default.
- A **soft** keepout allows non-relative-placement cells to come into its location during the legalization stage.
- A **space** keepout allows non-relative-placement cells to come into its location during placement.

DESCRIPTION

This command adds an item to the specified relative placement groups. The relative placement groups must have been previously defined by using the **create_rp_group** command. The item can be either a leaf cell, a relative placement group, or a keepout. The item is placed in a particular lattice position of the group as specified by a row and column.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **add_to_rp_group** to add a cell to an existing relative placement group and then adds that group hierarchically to another existing group:

```
prompt> get_rp_groups ripple::grp_ripple
{ripple::grp_ripple}

prompt> add_to_rp_group ripple::grp_ripple -leaf carry_in_1
{ripple::grp_ripple}

prompt> add_to_rp_group example3::top_group -hier grp_ripple -instance U2
{example3::top_group}

prompt> add_to_rp_group example3::top_group -leaf { U1 U2 U3 U4 U5 } \
        -row 0 -column 1 -num_rows 2 -num_columns 3 -free_placement
{example3::top_group}
```

SEE ALSO

create_rp_group(2)
 remove_rp_groups(2)
 write_rp_groups(2)

alias

Creates a pseudo-command that expands to one or more words, or lists current alias definitions.

SYNTAX

```
string alias  
  [name]  
  [def]
```

Data Types

```
name  string  
def   string
```

ARGUMENTS

name

Specifies a name of the alias to define or display. The name must begin with a letter, and can contain letters, underscores, and numbers.

def

Expands the alias. That is, the replacement text for the alias name.

DESCRIPTION

The **alias** command defines or displays command aliases. With no arguments, the **alias** command displays all currently defined aliases and their expansions. With a single argument, the **alias** command displays the expansion for the given alias name. With more than one argument, an alias is created that is named by the first argument and expanding to the remaining arguments.

You cannot create an alias using the name of any existing command or procedure. Thus, you cannot use **alias** to redefine existing commands.

Aliases can refer to other aliases.

Aliases are only expanded when they are the first word in a command.

EXAMPLES

Although commands can be abbreviated, sometimes there is a conflict with another command. The following example shows how to use **alias** to get around the conflict:

```
prompt> alias q quit
```

The following example shows how to use **alias** to create a shortcut for commonly-used command invocations:

```
prompt> alias include {source -echo -verbose}
```

```
prompt> alias rt100 {report_timing -max_paths 100}
```

After the previous commands, the command **include script.tcl** is replaced with **source -echo -verbose script.tcl** before the command is interpreted.

The following examples show how to display aliases using **alias**. Note that when displaying all aliases, they are in alphabetical order.

```
prompt> alias rt100
rt100 report_timing -max_paths 100
```

```
prompt> alias
include source -echo -verbose
q quit
rt100 report_timing -max_paths 100
```

SEE ALSO

unalias(2)

alib_analyze_libs

Reads in the target library files, analyzes each of them separately, and creates the corresponding alib files.

SYNTAX

integer **alib_analyze_libs**

DESCRIPTION

This command parses the *target_library* string. For each of the specified target libraries, the command attempts to find a valid alib. If no valid alib is found for target library X, library analysis occurs for X and the result is stored as X.alib in the **alib_library_analysis_path** variable directory. Note that the analysis is not affected by dont_use settings in the script where the command is issued.

EXAMPLES

Use the following command to analyze the x.db and y.db libraries:

```
set target_library "x.db y.db"  
set alib_library_analysis_path "./out"  
alib_analyze_libs
```

Two files, x.db.alib and y.db.alib are generated in a versioned directory under "./out". The directory is named "./out/alib-N", where N is the alib version number. To load these alibs, set the *alib_library_analysis_path* variable to ./out in your compile script. You do not need to specify the directory name, since this detail is managed entirely by the tool.

SEE ALSO

alib_library_analysis_path(3)
OPT-1310(n)
OPT-1311(n)

all_active_scenarios

Lists the active scenarios available in memory.

SYNTAX

string **all_active_scenarios**

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command displays all active scenarios currently in memory. This list excludes scenarios that are inactive.

Creation and usage of scenarios is available with Design Compiler Graphical.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following example uses the **all_active_scenarios** command to list the active scenarios.

```
prompt> create_scenario MODE1
prompt> create_scenario MODE2
prompt> create_scenario MODE3
prompt> set_active_scenarios {MODE1 MODE3}
prompt> all_active_scenarios
MODE1 MODE3
```

SEE ALSO

all_scenarios(2)
create_scenario(2)
current_scenario(2)
remove_scenario(2)
set_active_scenarios(2)

all_clock_gates

Returns a collection of clock-gating cells or pins in the current design.

SYNTAX

```
collection all_clock_gates
  [-no_hierarchy]
  [-clock clock_name]
  [-cells]
  [-enable_pins]
  [-clock_pins]
  [-output_pins]
  [-test_pins]
  [-observation_pins]
```

Data Types

clock_name string

ARGUMENTS

-no_hierarchy

Limits the search to only the current level of hierarchy. Subdesigns are not searched. By default, the search spans the full hierarchy down from the current level.

-clock *clock_name*

In the search, considers only clock-gating cells for registers originally clocked by *clock_name*. By default, the search considers registers clocked by all clocks.

-cells

Returns cells. This is the default behavior.

-enable_pins

Returns enable pins instead of cells.

-clock_pins

Returns clock input pins instead of cells.

-output_pins

Returns the output pins of the clock-gating cells that generate the gated clocks instead of the cells.

-test_pins

Returns test_mode or scan_enable input pins instead of cells.

-observation_pins

Returns the output pins of the observation logic, instead of cells.

DESCRIPTION

This command returns a collection of clock-gating cells or pins in the current instance, filtered by the specified options. By default, the command lists all clock-gating cells in the design. If you specify *clock_name*, the command lists only the clock-gating cells in the transitive fanout of the specified clock. The **-cells** option, which is the default, can be combined with one or more of the pins options.

XOR self-gating cells inserted by Power Compiler are excluded from the search. Use the **all_self_gates** command to get information about XOR self-gating cells.

EXAMPLES

The following example tags all clock gates for registers originally clocked by CLK for removal:

```
prompt> remove_clock_gating -gating_cells [all_clock_gates -clock CLK]
```

The following example returns a list of test pins for all clock gates for registers originally clocked by CLK:

```
prompt> all_clock_gates -test_pins -clock CLK
```

SEE ALSO

current_design(2)
current_instance(2)
report_clock_gating(2)

all_clocks

Returns a collection of all clocks in the current design.

SYNTAX

collection **all_clocks**

ARGUMENTS

This command has no arguments.

DESCRIPTION

Returns a collection containing all clocks in the current design. The clocks must be defined in the design before running this command.

To create clocks, use the **create_clock** command. To remove clocks, use the **remove_clock** command. To list detailed information about all clocks in the design, use the **report_clock** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example applies the **set_dont_touch_network** command to all clocks in the current design:

```
prompt> set_dont_touch_network [all_clocks]
```

SEE ALSO

`create_clock(2)`
`remove_clock(2)`

```
report_clock(2)  
set_dont_touch_network(2)
```

all_connected

Returns the objects connected to a net, port, pin, net instance, or pin instance.

SYNTAX

```
collection all_connected  
[-leaf]  
object
```

Data Types

object string

ARGUMENTS

-leaf

Specifies that only leaf pins are returned for a hierarchical net. For nonhierarchical nets, there is no difference in output.

object

Specifies the object whose connections are returned. The object must be a net, port, pin, net instance, or pin instance.

DESCRIPTION

The **all_connected** command returns a collection of objects connected to the specified net, port, pin, net instance, or pin instance. A net instance is a net in the hierarchy of the design. A pin instance is a pin on a cell in the hierarchy of a design.

If the **-leaf** option is used, a list of leaf pins of the net is returned.

To connect nets to ports or pins, use the **connect_net** command. To break connections, use the **disconnect_net** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **all_connected** to return the objects connected to MY_NET:

```
prompt> all_connected MY_NET
```

```
prompt> connect_net MY_NET OUT3
```

Connecting net 'MY_NET' to port 'OUT3'.

```
prompt> connect_net MY_NET U65/Z
```

Connecting net 'MY_NET' to pin 'U65/Z'.

```
prompt> all_connected MY_NET
```

```
{OUT3 U65/Z}
```

```
prompt> all_connected OUT3
```

```
{MY_NET}
```

```
prompt> all_connected U65/Z
```

```
{MY_NET}
```

This example uses **all_connected** to associate net load capacitance with the net n47, which is connected to the pin instance C/Z:

```
prompt> set_load 0.147 [all_connected [get_pins U0/U1/C/Z]]
```

Set "load" attribute to 0.147 for net "U0/U1/n47"

SEE ALSO

- all_inputs(2)
- all_outputs(2)
- connect_net(2)
- create_net(2)
- current_design(2)
- disconnect_net(2)
- remove_net(2)

all_critical_cells

Returns a collection of critical leaf cells in the top hierarchy of the current design.

SYNTAX

```
collection all_critical_cells
  [-slack_range range_value]
```

Data Types

range_value float

ARGUMENTS

-slack_range *range_value*

Specifies a margin of slack for searching top-hierarchy leaf cells in paths whose slacks are in the specified *range_value* relative to the worst slack of the current design. A top-hierarchy leaf cell is a cell in the top hierarchy and with no hierarchy underneath.

The *range_value* must be expressed in the same units as the technology library used during optimization. In addition, the *range_value* must be positive or 0.0. If no *range_value* is specified, the default is 0.0. A *range_value* of 0.0 means that top-hierarchy leaf cells in the most critical paths (those with the worst violations) are returned. If a positive *range_value* is specified, all top-hierarchy leaf cells are returned if they are in near-critical paths with slacks in the *range_value* relative to the worst slack of the current design.

DESCRIPTION

The **all_critical_cells** command returns a collection of leaf cells that are in the top hierarchy and in some path with a slack in the *range_value* relative to the worst slack of the current design. This command returns only those cells with no hierarchy underneath.

If all timing paths passing through a cell are unconstrained, the cell is assumed to be non-critical, and is not returned. You can use this command, along with the **group** command, to group together into a new subhierarchy those cells in the most critical paths. Then you can try various optimization techniques on the new subhierarchy.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example lists all top-hierarchy leaf cells in the worst critical paths of the current design:

```
prompt> all_critical_cells
```

The following example lists all top-hierarchy leaf cells in those paths within range 5.0 relative to the worst slack of the current design:

```
prompt> all_critical_cells -slack_range 5.0
```

The following example groups all top-hierarchy leaf cells in the worst critical paths into a new hierarchy, called CRIT, under the top hierarchy of the current design:

```
prompt> group [all_critical_cells] -design_name CRIT
```

The following example reports cell connections of all top-hierarchy leaf cells in the worst critical paths of the current design.

```
prompt> report_cell -connections [all_critical_cells]
```

SEE ALSO

`all_critical_pins(2)`

`current_design(2)`

`group(2)`

`report_cell(2)`

all_critical_pins

Returns a collection of critical endpoints or startpoints in the current design.

SYNTAX

```
collection all_critical_pins  
  [-type endpoint | startpoint]  
  [-slack_range range_value]
```

Data Types

range_value float

ARGUMENTS

-type endpoint | startpoint

Specifies that the pins or ports to be searched are either timing endpoints or timing startpoints. The default is **endpoint**.

-slack_range *range_value*

Specifies a margin of slack for searching timing endpoints (or startpoints) in paths whose slacks are in the specified *range_value* relative to the worst slack of the current design. The *range_value* must be expressed in the same units as the technology library used during optimization. In addition, the *range_value* must be positive or 0.0. If no *range_value* is specified, the default is 0.0. A *range_value* of 0.0 means that endpoints (or startpoints) in the most critical paths (those with the worst violations) will be returned. If a positive *range_value* is specified, endpoints (or startpoints) in near-critical paths with slacks in the *range_value* relative to the worst slack of the current design will be returned.

DESCRIPTION

The **all_critical_pins** command returns a collection of endpoints (or startpoints) that are in some timing path with a slack in the *range_value* relative to the worst slack of the current design. For a pin, if all timing paths passing through it are unconstrained, it is assumed to be non-critical, and it will not be returned.

This command may be used together with the **all_fanin** or **all_fanout** command to report useful information; for example, cells in the transitive timing fanin cone of the most critical endpoints.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example lists all endpoints in the worst critical paths of the current design:

```
prompt> all_critical_pins
```

The following example lists all startpoints in those paths within range 5.0 relative to the worst slack of the current design:

```
prompt> all_critical_pins -type startpoint -slack_range 5.0
```

The following example reports cells in the 3-level transitive fanin timing cone of endpoints in the worst critical paths:

```
prompt> report_cell [all_fanin -to [all_critical_pins] \  
-only_cells -levels 3 -flat]
```

The following example reports the logic in the transitive fanin of endpoints in the worst critical paths of the current design:

```
prompt> report_transitive_fanin -to [all_critical_pins]
```

SEE ALSO

- all_critical_cells(2)
- all_fanin(2)
- current_design(2)
- group(2)
- report_cell(2)
- report_transitive_fanin(2)
- report_transitive_fanout(2)

all_designs

Returns a collection containing all designs in the current design.

SYNTAX

collection **all_designs**

ARGUMENTS

There are no arguments to this command.

DESCRIPTION

The **all_designs** command returns a collection containing the designs in the current design hierarchy in bottom-up order. You must set the current design using the **current_design** command before using **all_designs**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

current_design(2)

all_dont_touch

Returns a collection of **dont_touch** cells or nets from the current design or from the specified input collection.

SYNTAX

```
collection all_dont_touch  
-cells | -nets  
[input_coll]
```

Data Types

input_coll string

ARGUMENTS

-cells

Specifies that all cells with the **dont_touch** attribute are to be returned. The **-cells** and **-nets** options are mutually exclusive.

-nets

Specifies that all nets with the **dont_touch** attribute are to be returned. The **-cells** and **-nets** options are mutually exclusive.

input_coll

Searches the specified input collection and returns the collection of cells or nets having the **dont_touch** attribute. Objects are to be searched only from the content of *input_coll* rather than from the entire design. By default, this option is off.

DESCRIPTION

This command returns the collection of cells or nets that have the **dont_touch** attribute from the design or from a list that you specify. You can use wildcard characters, such as an asterisk (*) or question mark (?), when specifying the input cell list.

The command returns a standard Tcl collection. You can perform all of the generic collection-manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, on this collection handle.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows the collection of all cells with the **dont_touch** attribute in the design:

```
prompt> all_dont_touch -cells  
{U20 U21 U23 SBLK/U1 SBLK/U2}
```

The following example shows the collection of all nets with the **dont_touch** attribute from an existing collection where *COLL* is its handle:

```
prompt> all_dont_touch -nets $COLL  
{n2 n5 SBLK/n2}
```

SEE ALSO

```
foreach_in_collection(2)  
get_attribute(2)  
get_cells(2)  
get_nets(2)  
report_attribute(2)  
set_attribute(2)  
set_dont_touch(2)  
set_dont_touch_network(2)  
sizeof_collection(2)
```

all_drc_violated_nets

Returns a collection of DRC-violated nets from the current design or from the specified input collection.

SYNTAX

```
collection all_drc_violated_nets  
  -max_capacitance | -max_transition | -max_fanout  
  [input_coll]  
  [-bound upper]  
  [-threshold threshold]
```

Data Types

<i>input_coll</i>	collection
<i>upper</i>	float
<i>threshold</i>	float

ARGUMENTS

-max_capacitance

Returns the collection of maximum capacity nets that violate design rule checking (DRC), as designated by **drc_violated_nets**.

-max_transition

Returns the collection of maximum transition nets that violate DRC, as designated by **drc_violated_nets**.

-max_fanout

Returns the collection of maximum fanout nets that violate DRC, as designated by **drc_violated_nets**.

input_coll

Searches the specified input collection and returns the requested collection of nets that violate DRC constraints. Objects are searched only from the contents of the specified input collection, rather than from the design.

-bound *upper*

Captures all DRC-violated nets that have values less than or equal to the bound specified by *upper*. By default, this option is off.

-threshold *threshold*

Captures all DRC-violated nets that have values greater than or equal to the threshold specified by *threshold*. By default, this option is off.

DESCRIPTION

The **all_drc_violated_nets** command returns a collection of DRC-violated nets from the current design or from the specified input collection. The tool can search the entire design hierarchy. It returns DRC-violated nets from the entire design, not just from the top level.

This command returns the 3 most common types of DRC violations: **max_capacitance**, **max_transition**, and **max_fanout**. You can specify each of the violations separately in this command. If you do not specify an option, a collection of all three types of DRC-violated nets is returned.

The command returns a standard Tcl collection. You can perform all of the generic collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, on this collection handle.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows the collection of all maximum capacity DRC-violated nets from the design:

```
prompt> all_drc_violated_nets -max_cap  
{n1 n2 n3 abc/n14}
```

The following example shows the collection of all **-max_fanout** DRC-violated nets from an existing collection stored in \$COLL:

```
prompt> all_drc_violated_nets -max_fanout $COLL  
{n1 n3 abc/n14}
```

SEE ALSO

`foreach_in_collection(2)`
`sizeof_collection(2)`

all_fanin

Reports pins, ports, or cells in the fanin of specified sinks.

SYNTAX

```
collection all_fanin  
  -to sink_list  
  [-startpoints_only]  
  [-exclude_bboxes]  
  [-break_on_bboxes]  
  [-only_cells]  
  [-flat]  
  [-levels count]  
  [-trace_arcs arc_type]
```

Data Types

```
sink_list  list  
count      int
```

ARGUMENTS

-to *sink_list*

Reports a list of sink pins, ports, or nets in the design and a timing fanin of each sink in the *sink_list*. If you specify a net, the effect is the same as listing all driver pins on the net.

-startpoints_only

Returns only the timing startpoints.

-exclude_bboxes

Excludes black boxes from the final result.

-break_on_bboxes

Stops timing fanin traversal on black boxes.

-only_cells

Results in a set of all cells in the timing fanin of the *sink_list*.

-flat

Specifies to function in the flat mode of operation. The two major modes in which **all_fanin** functions are hierarchical (the default)

and flat. When in hierarchical mode, only objects from the same hierarchy level as the current sink are returned. Thus, pins within a level of hierarchy lower than that of the sink are used for traversal but are not reported.

-levels *count*

Stops traversal when reaching the perimeter of the search of *count* hops, where counting is performed over the layers of cells that are equidistant from the sink.

-trace_arcs *arc_type*

Specifies the type of combinational arcs to trace during the traversal. Allowed values are **timing**, which permits the tracing only of valid timing arcs (arcs that are neither disabled nor invalid due to case analysis), and **all**, which permits tracing of all combinational arcs regardless of either case analysis or arc disabling. The default in Design Compiler (both in topographical mode and non-topographical mode) is **timing**. The default in DC Explorer is **all**. You can change the default by setting the **fanin_fanout_trace_arcs** variable to the desired value.

DESCRIPTION

The **all_fanin** command reports the timing fanin of specified sink pins, ports, or nets in the design. A pin is considered to be in the timing fanin of a sink if there is a timing path through combinational logic from the pin to that sink. The fanin report stops at the clock pins of registers (sequential cells).

Multicorner-Multimode Support

Depending on the options used, this command either uses the current scenario or has no dependency on scenario-specific information.

EXAMPLES

The following examples show the timing fanin of a port in the design. The design comprises three inverters in a chain named *iv1*, *iv2*, and *iv3*. The *iv1* and *iv2* inverters are hierarchically combined in a larger cell named *ii2*.

```
prompt> all_fanin -to tout
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

```
prompt> all_fanin -to tout -flat
{ii2/iv1/U1/a ii2/iv2/U1/z tin iv3/U1/a ii2/iv1/U1/z
ii2/iv2/U1/a iv3/U1/z tout}
```

SEE ALSO

`all_fanout(2)`
`report_transitive_fanin(2)`

all_fanout

Returns a set of pins, ports, or cells in the fanout of the specified sources.

SYNTAX

```
collection all_fanout
  -clock_tree
  -from source_list
  [-endpoints_only]
  [-exclude_bboxes]
  [-break_on_bboxes]
  [-only_cells]
  [-flat]
  [-levels count]
  [-trace_arcs arc_type]
```

Data Types

```
source_list  list
count        int
```

ARGUMENTS

-clock_tree

Uses all clock source pins and/or ports in the design as the list of sources. Clock sources are specified by using the **create_clock** command. If there are no clocks, or if the clocks have no sources, the report is empty. Use the **report_clock** command to list the sources for all clocks in the design. The **-clock_tree** option generates a report that displays the clock trees or networks in the design. The **-clock_tree** and **-from** options are mutually exclusive.

-from *source_list*

Specifies a list of source pins, ports, or nets in the design. The timing fanout of each source in the *source_list* is reported. If a net is specified, the effect is the same as listing all load pins on the net. The **-clock_tree** and **-from** options are mutually exclusive.

-endpoints_only

Returns only timing endpoints as a result.

-exclude_bboxes

Excludes black boxes from the final result.

-break_on_bboxes

Stops timing fanout traversal on black boxes.

-only_cells

Results in a set of all cells in the timing fanout of the *source_list*, rather than a set of pins or ports.

-flat

Specifies to function in the flat mode of operation. The two major modes in which **all_fanout** functions are hierarchical (the default) and flat. When in hierarchical mode, only objects from the same hierarchy level as the current source are returned. Thus, pins within a level of hierarchy lower than that of the source are used for traversal but are not reported.

-levels count

Stops traversal when reaching the perimeter of the search of *count* hops, where counting is performed over the layers of cells that are equidistant from the source.

-trace_arcs arc_type

Specifies the type of combinational arcs to trace during the traversal. Allowed values are **timing**, which permits the tracing only of valid timing arcs (arcs that are neither disabled nor invalid due to case analysis), and **all**, which permits tracing of all combinational arcs regardless of either case analysis or arc disabling. The default in IC Compiler and Design Compiler (both in topographical mode and non-topographical mode) is **timing**. The default in DC Explorer is **all**. The default can be changed by setting the **fanin_fanout_trace_arcs** variable to the desired value.

DESCRIPTION

The **all_fanout** command reports the timing fanout of specified source pins, ports, or nets in the design. A pin is considered to be in the timing fanout of a sink if there is a timing path through combinational logic from that source to the pin. The fanout report stops at the inputs to registers (sequential cells). The source pins or ports are specified by using the **-clock_tree** or **-from source_list** option.

Multicorner-Multimode Support

Depending on the options used, this command either uses the current scenario or has no dependency on scenario-specific information.

EXAMPLES

The following example shows the timing fanout of a port in the design. The design comprises the following three inverters in a chain named *iv1*, *iv2*, and *iv3*. The *iv1* and *iv2* inverters are hierarchically combined in a larger cell named *ii2*.

```
prompt> all_fanout -from tin
{iv3/out tout iv3/in ii2/hin ii2/hout tin}
```

```
prompt> all_fanout -from tin -flat
{tout ii2/iv2/U1/z ii2/iv1/U1/a iv3/U1/z iv3/U1/a
ii2/iv2/U1/a ii2/iv1/U1/z tin}
```

```
prompt> all_fanout -from tin -levels 1 -only_cells
{iv3 ii2}
```

SEE ALSO

`all_fanin(2)`
`create_clock(2)`
`report_clock(2)`
`report_transitive_fanout(2)`

all_high_fanout

Returns a collection of high-fanout nets from the current design or from the specified input collection.

SYNTAX

```
collection all_high_fanout  
-nets  
[-threshold value]  
[input_coll]  
[-through_buf_inv]
```

Data Types

<i>value</i>	float
<i>input_coll</i>	collection

ARGUMENTS

-nets

Returns the collection of high-fanout nets.

-threshold *value*

Specifies a threshold value used to determine if a net is a high-fanout net. The *value* is a user-specified value. By default, the **high_fanout_net_threshold** variable value is used to determine if a net is a high-fanout net.

input_coll

Searches the specified input collection for high-fanout nets. Objects are to be searched only from the contents of the specified input collection, rather than from the design.

-through_buf_inv

Indicates to treat a buffer tree as transparent. The leaf loads of the buffer tree are treated as the fanouts of the net.

DESCRIPTION

The **all_high_fanout** command returns a collection of all high-fanout nets in the design. The tool searches the entire design hierarchy. It returns nets from the entire design, not just from the top level.

To determine if a net is a high-fanout net, use the **high_fanout_net_threshold** variable value as the threshold value. The command

returns all nets with a fanout count higher than this threshold as high-fanout nets. You can also specify the threshold by using the **-threshold** option.

If you specify a value for *input_coll*, the command searches for high-fanout objects only from the specified collection, rather than from the entire design.

A standard Tcl collection is returned. All of the generic collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, can be performed on this collection handle.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the collection of all high-fanout nets from the design:

```
prompt> all_high_fanout -nets  
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

The following example shows the collection of all high-fanout nets from an existing collection stored in \$COLL:

```
prompt> all_high_fanout -nets $COLL  
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

The following example shows the collection of all high-fanout nets from the design having a fanout count of more than 100:

```
prompt> all_high_fanout -nets -threshold 100  
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

SEE ALSO

all_fanin(2)
all_fanout(2)
foreach_in_collection(2)
report_net_fanout(2)
sizeof_collection(2)

all_ideal_nets

Returns a collection of ideal nets from the current design or from the specified input collection.

SYNTAX

```
collection all_ideal_nets
    [input_coll]
```

Data Types

input_coll collection

ARGUMENTS

input_coll

Specifies the input collection to search for ideal nets.

If you do not specify this argument, the command searches for ideal nets in the current design.

DESCRIPTION

The **all_ideal_nets** command returns a collection of ideal nets.

If you specify a value for *input_coll*, the command searches for ideal nets only in the specified collection.

If you do not specify a collection, the command searches for ideal nets in the current design. The tool searches the entire design hierarchy. The command returns ideal nets from the entire design, not just from the top level.

A standard Tcl collection is returned. All of the generic collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, can be performed on this collection handle.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns a collection of all ideal nets from the design:

```
prompt> all_ideal_nets  
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

The following example returns a collection of all ideal nets from an existing collection stored in \$COLL:

```
prompt> all_ideal_nets $COLL  
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

SEE ALSO

```
all_fanout(2)  
foreach_in_collection(2)  
sizeof_collection(2)
```

all_inputs

Returns a collection of input or inout ports in the current design.

SYNTAX

```
collection all_inputs
  [-clock clock_name]
  [-edge_triggered | -level_sensitive]
```

Data Types

clock_name string

ARGUMENTS

-clock *clock_name*

Limits the search to ports that have input delay relative to *clock_name*.

-edge_triggered

Limits the search to ports that have edge-triggered input delay as specified by the **set_input_delay** command with the **-clock** option.

-level_sensitive

Limits the search to ports that have level-sensitive input delay as specified by the **set_input_delay** command with the **-level_sensitive** option.

DESCRIPTION

The **all_inputs** command returns a collection of all input or inout ports in the current design, unless one of the options limits the search. The **all_inputs** command is usually used with a command that places attributes on input ports. To get detailed information on ports in the current design, use the **report_port** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example lists all input ports in the current design:

```
prompt> all_inputs  
{A1 A2 BIDIR1}
```

The following example sets the drive value of all the input ports on the current design to 10:

```
prompt> set_drive 10.0 [all_inputs]
```

The following example marks with a multicycle value of 0 all paths from inputs having level-sensitive input delay relative to PHI1 to level-sensitive registers clocked by PHI1:

```
prompt> set_multicycle_path 0 \  
      -from [all_inputs -clock PHI1 -level_sensitive] \  
      -to [all_registers -data_pins -clock PHI1]
```

SEE ALSO

- `all_outputs(2)`
- `current_design(2)`
- `report_port(2)`
- `set_drive(2)`
- `set_input_delay(2)`
- `set_multicycle_path(2)`

all_isolation_cells

Returns a collection of isolation cells available in the design.

SYNTAX

collection **all_isolation_cells**

ARGUMENTS

The **all_isolation_cells** command has no arguments.

DESCRIPTION

The **all_isolation_cells** command returns a collection of the existing isolation cells in the design. The enabled level shifters also work as isolation cells, but that type of cell is not returned by this command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

all_level_shifters(2)
check_mv_design(2)

all_level_shifters

Returns a collection of level-shifter cells available in the design.

SYNTAX

```
collection all_level_shifters  
[-type els | simple]
```

ARGUMENTS

-type els | simple

Specifies the type of level-shifter cells to include in the collection. If you specify **-type els**, the command returns only the enabled level-shifter cells. If you specify **-type simple**, the command returns only regular level-shifter cells. If you do not specify this option, the command returns all level-shifter cells.

DESCRIPTION

The **all_level_shifters** command returns a collection of existing level-shifter cells in the design. You can restrict the types of level-shifter cells returned in the collection by using the **-type** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

all_macro_cells

Returns a collection of all macro cells in the design or from a list of objects in the input collection.

SYNTAX

```
collection all_macro_cells
    [input_coll]
```

Data Types

input_coll collection

ARGUMENTS

input_coll

Specifies the input collection to search and return the collection of macro cells. Objects are to be searched only from the content of the specified input collection rather than from the design.

DESCRIPTION

This command can return the collection of all macro cells from the design. The tool searches the entire design hierarchy. It returns cells from the entire design, not just from the top level. This command uses the same definition as used in the legalizing process to determine if a cell is a macro cell.

If you specify a value for *input_coll*, the command searches for macro cells only from the specified collection, rather than from the entire design.

A standard Tcl collection is returned. All of the generic collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, can be performed on this collection.

A library cell can be set as macro cell by using `set_cell_type` command. If user requires any cell to be set as macro cell, `cell_type` needs to be made BLOCK, COVER or RING. These `cell_type` attributes are the `lef` attributes and coming from LEF on all the existing macro cells. To remove these macro cell attributes from any macro library cell, `cell_type` can be set as PAD or CORE type.

In case of non-topographical mode, the all macro cells count will be same as the "Macro Count" results of the `report_qor`.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the collection of all macro cells from the design.

```
prompt> all_macro_cells  
{macro1 top/macro2 top/macro4 macro10}
```

The following example shows the collection of all macro_cells from an existing collection stored in \$COLL.

```
prompt> all_macro_cells $COLL  
{macro1 macro2}
```

SEE ALSO

```
foreach_in_collection(2)  
sizeof_collection(2)
```

all_outputs

Returns a collection of output or inout ports in the current design.

SYNTAX

```
collection all_outputs
  [-clock clock_name]
  [-edge_triggered | -level_sensitive]
```

Data Types

clock_name string

ARGUMENTS

-clock *clock_name*

Limits the search to ports that have output delay relative to *clock_name*.

-edge_triggered

Limits the search to ports that have edge-triggered output delay as specified by the **set_output_delay** command with the **-clock** option.

-level_sensitive

Limits the search to ports that have level-sensitive output delay as specified by the **set_output_delay** command with the **-level_sensitive** option.

DESCRIPTION

The **all_outputs** command returns a collection of all output or inout ports in the current design, unless one of the options limits the search. This command is usually used with a command that places attributes on output ports. To get detailed information on ports in the current design, use the **report_port** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example lists all output ports:

```
prompt> all_outputs  
{OUT1 OUT2 BIDIR1}
```

The following example sets a capacitive load of 3.5 on each output port:

```
prompt> set_load 3.5 [all_outputs]
```

The following example sets paths leading to output ports clocked by TSTCLK to be false.

```
prompt> set_false_path -to [all_outputs -clock TSTCLK]
```

SEE ALSO

- all_inputs(2)
- current_design(2)
- report_port(2)
- set_false_path(2)
- set_load(2)
- set_output_delay(2)

all_physical_only_cells

Returns a collection of all the physical-only cells from a design or from a list of objects in an input collection.

SYNTAX

```
collection all_physical_only_cells  
  [-lib_cells lib_list | -cell_name list]  
  [-coordinates {llx lly urx ury}]  
  [input_coll_handle]
```

Data Types

<i>lib_list</i>	list
<i>list</i>	list
<i>llx</i>	float
<i>lly</i>	float
<i>urx</i>	float
<i>ury</i>	float
<i>input_coll_handle</i>	collection

ARGUMENTS

-lib_cells *lib_list*

Specifies a list of library cells. The tool searches for physical-only cells in the specified library cell list. The default behavior returns all physical-only cells from the entire design.

-cell_name *list*

Specifies the base instance name of the filler cells to be searched for in the design. The default behavior returns all physical-only cells from the entire design.

-coordinates {*llx lly urx ury*}

Specifies the coordinates of the area to search for physical-only cells. Returns a collection of the physical-only cells contained within the specified area. The default behavior returns all physical-only cells from the entire design.

input_coll_handle

Specifies the name of a collection from which to search for physical-only cells. Returns a collection of physical-only cells that are contained within the specified input collection. The default behavior returns all physical-only cells from the entire design.

DESCRIPTION

This command returns a collection of all the physical-only cells in the design. The tool searches the entire design hierarchy. It returns physical-only cells from the entire design, not just from the top level.

You can use wild cards, such as an asterisk (*) or question mark (?), when specifying cell instance names.

A standard Tcl collection is returned. All of the collection-manipulating commands, such as the **foreach_in_collection** and **sizeof_collection** commands, can be performed on the collection of physical-only cells.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example collects all the physical-only cells from the design.

```
prompt> all_physical_only_cells
{fill1 fill2 fill3 abc/fill4}
```

The following example collects all the physical-only cells from the design that have a library cell of type *OKOAFILLER*.

```
prompt> all_physical_only_cells -lib_cell OKOAFILLER
{hin in out tin hout tout}
```

The following example collects all the physical-only cells from an existing collection stored in *\$COLL*.

```
prompt> all_physical_only_cells $COLL
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

The following example collects all the physical-only cells from the design that have an instance name starting with *fill*.

```
prompt> all_physical_only_cells -cell_name fill*
{fill1 fill2 fill3}
```

SEE ALSO

`foreach_in_collection(2)`
`sizeof_collection(2)`

all_registers

Returns a collection of sequential cells or pins in the current design.

SYNTAX

```
collection all_registers
  [-no_hierarchy]
  [-clock clock_name]
  [-rise_clock rise_clock_name]
  [-fall_clock fall_clock_name]
  [-cells]
  [-data_pins]
  [-clock_pins]
  [-slave_clock_pins]
  [-output_pins]
  [-level_sensitive | -edge_triggered]
  [-master_slave]
  [-include_icg]
```

Data Types

```
clock_name      string
rise_clock_name string
fall_clock_name string
```

ARGUMENTS

-no_hierarchy

Limits the search to only the current level of hierarchy. Subdesigns are not searched.

By default, the entire hierarchy is searched.

-clock *clock_name*

Considers only sequential cells clocked by the specified clock.

By default, all sequential cells in the current design are considered.

-rise_clock *rise_clock_name*

Considers only sequential cells triggered by the rising edge of the specified clock.

By default, all sequential cells in the current design are considered.

-fall_clock *fall_clock_name*

Considers only sequential cells triggered by the falling edge of the specified clock.

By default, all sequential cells in the current design are considered.

-cells

Returns a collection of sequential cells that meet the search criteria.

If you do not specify any of the object type options, the command returns a collection of sequential cells.

-data_pins

Returns a collection of data pins of the sequential cells that meet the search criteria.

-clock_pins

Returns a collection of clock pins of the sequential cells that meet the search criteria.

-slave_clock_pins

Returns a collection of slave clock pins of master-slave registers that meet the search criteria. Slave clock pins are specified as `clocked_on_also` in the library.

-output_pins

Returns a collection of output pins of the sequential cells that meet the search criteria.

-level_sensitive

Limits the search to level-sensitive cells.

-edge_triggered

Limits the search to edge-triggered cells.

-master_slave

Limits search to master_slave cells.

-include_icg

Returns a collection containing integrated clock-gating cells.

DESCRIPTION

The **all_registers** command returns a collection of sequential cells or pins in the current design, filtered as specified by the options. By default, the command returns a collection of all sequential cells in the design. If you specify *clock_name*, it considers only the sequential cells in the transitive fanout of the sources of the clock.

You can use one or more of the **-cells**, **-data_pins**, **-clock_pins**, **-slave_clock_pins**, and **-output_pins** options to return a collection containing the respective types of objects. For example, if you use **-data_pins**, the command returns a collection containing the only the data pins of the sequential cells that meet the search criteria. If you use both **-cells** and **-data_pins**, the command returns a collection containing both the sequential cells and their data pins.

When the variable **all_registers_include_icg** is set to TRUE, the command returns a collection containing integrated clock-gating cells.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets `max_delay` targets for timing paths leading to data pins of all registers clocked by *PHI2*:

```
prompt> set_max_delay 10.0 -to [all_registers -clock PHI2 -data_pins]
```

The following example returns a list of data pins for all master-slave registers clocked by *clockB*:

```
prompt> all_registers -master_slave -data_pins -clock clockB
```

The following example returns a list of all level-sensitive cells and their clock (enable) pins:

```
prompt> all_registers -level_sensitive -cells -clock_pins
```

The following example shows how to push into an instance named *U1* and find level-sensitive cells without searching subdesigns of that instance:

```
prompt> current_instance U1
```

```
prompt> all_registers -no_hierarchy
```

SEE ALSO

- `create_clock(2)`
- `current_design(2)`
- `current_instance(2)`
- `remove_clock(2)`
- `set_max_delay(2)`
- `all_registers_include_icg(3)`

all_rp_groups

Returns a collection of specified relative placement groups and all groups in their hierarchy.

SYNTAX

collection **all_rp_groups**
[*rp_groups*]

Data Types

rp_groups list or collection

ARGUMENTS

rp_groups

Specifies the relative placement groups for which to search.

If you do not specify this argument, the command returns a collection that contains all relative placement groups currently loaded in memory.

DESCRIPTION

The **all_rp_groups** command returns a collection of the specified groups and all subgroups in their hierarchy.

If you do not specify the *rp_groups* argument, the command returns all relative placement groups currently loaded in memory.

If no relative placement groups are found, the command returns an empty string.

This command is supported only for designs that do not contain multiply-instantiated designs.

Relative placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **all_rp_groups** command:

```
prompt> get_rp_groups  
{mul::grp_mul ripple::grp_ripple example3::top_group}
```

```
prompt> all_rp_groups  
{mul::grp_mul ripple::grp_ripple example3::top_group}
```

```
prompt> all_rp_groups ripple::grp_ripple  
{ripple::grp_ripple mul::grp_mul}
```

```
prompt> remove_rp_groups -all -quiet  
1
```

```
prompt> all_rp_groups
```

SEE ALSO

- add_to_rp_group(2)
- all_rp_hierarchicals(2)
- all_rp_inclusions(2)
- all_rp_instantiations(2)
- all_rp_references(2)
- create_rp_group(2)
- remove_rp_groups(2)

all_rp_hierarchicals

Returns a collection of hierarchical relative placement groups that contain the specified groups in their hierarchy. The specified groups can be either included or instantiated in their parent group.

SYNTAX

```
collection all_rp_hierarchicals  
  [rp_groups]
```

Data Types

rp_groups list or collection

ARGUMENTS

rp_groups

Specifies the relative placement groups whose ancestor groups are to be in the collection returned by this command. The specified groups can be either included or instantiated in their parent group.

If you do not specify this argument, this command returns a collection that contains all hierarchical relative placement groups currently loaded in memory.

DESCRIPTION

The **all_rp_hierarchicals** command returns a collection of hierarchical groups that are ancestors of the relative placement groups specified in the *rp_groups* argument.

If you do not specify the *rp_groups* argument, the command returns a collection that contains all hierarchical relative placement groups currently loaded in memory.

If no relative placement groups are found, the command returns an empty string.

This command is supported only for designs that do not contain multiply-instantiated designs.

Relative placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **all_rp_hierarchicals** command:

```
prompt> get_rp_groups
{b::top a0::a0_group b::u_top/a1_group
 b::u_top/a1_group_include b::u_nxt/a1_group
 b::a1_group_include a1::a1_group}

prompt> all_rp_hierarchicals
{b::a1_group_include b::u_nxt/a1_group
 b::u_top/a1_group_include b::u_top/a1_group b::top}

prompt> all_rp_hierarchicals b::u_top/a1_group_include
{b::u_top/a1_group}

prompt> remove_rp_groups -all -quiet
1

prompt> all_rp_hierarchicals
```

SEE ALSO

- add_to_rp_group(2)
- all_rp_groups(2)
- all_rp_inclusions(2)
- all_rp_instantiations(2)
- all_rp_references(2)
- create_rp_group(2)
- remove_rp_groups(2)

all_rp_inclusions

Returns a collection that contains the hierarchical relative placement groups that include the specified groups.

SYNTAX

collection **all_rp_inclusions**
[*rp_groups*]

Data Types

rp_groups list or collection

ARGUMENTS

rp_groups

Specifies the included relative placement groups whose parent groups are to be included in the collection returned by this command.

If you do not specify this argument, this command returns a collection that contains all hierarchical relative placement groups in the design that contain included groups.

DESCRIPTION

The **all_rp_inclusions** command returns a collection that contains the hierarchical groups that include the relative placement groups specified in the *rp_groups* argument.

If you do not specify the *rp_groups* argument, the command returns a collection that contains all hierarchical relative placement groups currently in memory that include other relative placement groups.

If no relative placement groups are found, the command returns an empty string.

This command is supported only for designs that do not contain multiply-instantiated designs.

Relative placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **all_rp_inclusions** command:

```
prompt> get_rp_groups  
{mul::grp_mul mul::big_grp ripple::grp_ripple  
example3::g2 example3::top_group}
```

```
prompt> all_rp_inclusions  
{mul::big_grp example3::g2}
```

```
prompt> all_rp_inclusions mul::grp_mul  
{mul::big_grp}
```

```
prompt> remove_rp_groups -all -quiet  
1
```

```
prompt> all_rp_inclusions
```

SEE ALSO

```
add_to_rp_group(2)  
all_rp_groups(2)  
all_rp_hierarchicals(2)  
all_rp_instantiations(2)  
all_rp_references(2)  
create_rp_group(2)  
remove_rp_groups(2)
```

all_rp_instantiations

Returns a collection of hierarchical relative placement groups that instantiate the specified groups.

SYNTAX

collection **all_rp_instantiations**
[*rp_groups*]

Data Types

rp_groups list or collection

ARGUMENTS

rp_groups

Specifies the instantiated relative placement groups whose parent groups are to be included in the collection returned by this command.

If you do not specify this argument, this command returns a collection that contains all hierarchical relative placement groups in the design that contain instantiated groups.

DESCRIPTION

The **all_rp_instantiations** command returns a collection of hierarchical relative placement groups that instantiate the groups specified in the *rp_groups* argument.

If you do not specify the *rp_groups* argument, the command returns a collection that contains all hierarchical relative placement groups currently in memory that instantiate other relative placement groups.

If no relative placement groups are found, the command returns an empty string.

This command is supported only for designs that do not contain multiply-instantiated designs.

Relative placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **all_rp_instantiations** command:

```
prompt> get_rp_groups  
{mul::grp_mul ripple::grp_ripple example3::top_group}
```

```
prompt> all_rp_instantiations  
{ripple::grp_ripple example3::top_group}
```

```
prompt> all_rp_instantiations ripple::grp_ripple  
{example3::top_group}
```

```
prompt> remove_rp_groups -all -quiet  
1
```

```
prompt> all_rp_instantiations
```

SEE ALSO

```
add_to_rp_group(2)  
all_rp_groups(2)  
all_rp_hierarchicals(2)  
all_rp_instantiations(2)  
all_rp_references(2)  
create_rp_group(2)  
remove_rp_groups(2)
```

all_rp_references

Returns a collection of relative placement groups that directly contain the specified cells, which are either leaf cells or hierarchical cells that contain instantiated relative placement groups.

SYNTAX

```
collection all_rp_references
  [cell_list]
  [-design design_name]
```

Data Types

```
cell_list    list
design_name  string
```

ARGUMENTS

cell_list

Specifies the cells (either leaf cells or hierarchical cells that contain instantiated relative placement groups) whose parent groups are to be included in the collection returned by this command.

If you do not specify this argument, the command returns a collection that contains all relative placement groups that directly contain any leaf cells in the specified design.

-design *design_name*

Specifies the design in which to locate the specified cells.

If you do not specify this option, the tool looks for the cells in the current design.

DESCRIPTION

The **all_rp_references** command returns a collection of relative placement groups that directly contain the cells (either leaf cells or hierarchical cells that contain instantiated relative placement groups) specified in the *cell_list* argument.

If you do not specify the *cell_list* argument, the command returns a collection that contains all relative placement groups that directly contain any leaf cells in the specified design.

A group directly contains a leaf cell if the cell was added to the group by using the **-leaf** option of the **add_to_rp_group** command.

A group directly contains a hierarchical cell if the cell was used to hierarchically instantiate another group by using the **-instance**

option of the **add_to_rp_group** command.

This command is supported only for designs that do not contain multiply-instantiated designs.

Relative placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **all_rp_references** command:

```
prompt> get_rp_groups  
{mul::grp_mul ripple::grp_ripple example3::ripple  
example3::top_group}
```

```
prompt> all_rp_references  
{mul::grp_mul ripple::grp_ripple example3::ripple}
```

```
prompt> all_rp_references U1 -design mul  
{mul::grp_mul}
```

```
prompt> remove_rp_groups -all -quiet  
1
```

```
prompt> all_rp_references
```

SEE ALSO

- `add_to_rp_group(2)`
- `all_rp_groups(2)`
- `all_rp_inclusions(2)`
- `all_rp_instantiations(2)`
- `create_rp_group(2)`
- `remove_rp_groups(2)`

all_scenarios

Lists all defined scenarios available in memory.

SYNTAX

string **all_scenarios**

ARGUMENTS

The **all_scenarios** command has no arguments.

DESCRIPTION

The **all_scenarios** command displays the scenarios currently defined in memory. This list includes both active and inactive scenarios.

Creation and usage of scenarios is available with Design Compiler Graphical.

Multicorner-Multimode Support

This command uses information from both active and inactive scenarios.

EXAMPLES

The following example uses **all_scenarios** to list the available scenarios:

```
prompt> create_scenario MODE1
```

```
prompt> create_scenario MODE2
```

```
prompt> all_scenarios  
MODE1 MODE2
```

SEE ALSO

all_active_scenarios(2)
create_scenario(2)
current_scenario(2)
remove_scenario(2)
set_active_scenarios(2)

all_self_gates

Returns a collection of self-gating cells or pins in the current design. This command is supported only in topographical mode.

SYNTAX

```
collection all_self_gates
  [-no_hierarchy]
  [-clock clock_name]
  [-cells]
  [-enable_pins]
  [-clock_pins]
  [-output_pins]
  [-test_pins]
```

Data Types

clock_name string

ARGUMENTS

-no_hierarchy

Limits the search to only the current level of hierarchy. Subdesigns are not searched. By default, this option is off.

-clock *clock_name*

Considers only self-gating cells for registers originally clocked by *clock_name* in the search. The *clock_name* can be either a string or collection of one object. By default, this option is off.

-cells

Returns cells. This is the default.

-enable_pins

Returns enable pins instead of cells. By default, this option is off.

-clock_pins

Returns clock-input pins instead of cells. By default, this option is off.

-output_pins

Returns gated-clock output pins instead of cells. By default, this option is off.

-test_pins

Returns test_mode or scan_enable input pins instead of cells. By default, this option is off.

DESCRIPTION

This command returns a collection of self-gating cells or pins in the current design, filtered as specified by the options. By default, the command lists self-gating cells in the design. If you specify *clock_name*, it considers self-gating cells in the transitive fanout of the specified clock in the search. Normally, it considers all self-gating cells in the search. The **-cells** option, which is the default, can be combined with one or more of the pins options.

Traditional clock-gating cells inserted by the Power Compiler are excluded from the search. Use the **all_clock_gates** command to get information about them.

EXAMPLES

The following example returns all self-gating cells for registers originally clocked by CLK:

```
prompt> all_self_gates -clock CLK
```

The following example returns a list of test pins for all self-gating cells for registers originally clocked by CLK:

```
prompt> all_self_gates -test_pins -clock CLK
```

SEE ALSO

compile_ultra(2)
report_self_gating(2)
all_clock_gates(2)

all_test_modes

Displays all of the test modes that are defined for the current design.

SYNTAX

```
list all_test_modes
```

ARGUMENTS

The **all_test_modes** command has no arguments.

DESCRIPTION

The **all_test_modes** returns a list of all the modes defined for the current design. The **define_test_mode** command is used to define a test mode of operation for a design.

EXAMPLES

The following is an example of the test modes report:

```
set mode_list [all_test_modes]
foreach m $mode_list {
  current_test_mode $m
  dft_drc
}
```

SEE ALSO

```
list_test_modes(2)
current_design(2)
current_test_mode(2)
define_test_mode(2)
```

```
insert_dft(2)
```

all_threestate

Returns a collection of three-state cells or nets.

SYNTAX

```
collection all_threestate  
-nets  
[input_coll]
```

Data Types

input_coll collection

ARGUMENTS

-nets

Specifies that the collection of three-state nets is to be returned.

input_coll

Searches the specified input collection and returns a collection of three-state nets. Objects are to be searched only from the content of the specified input collection, rather than from the design.

DESCRIPTION

The **all_threestate** command returns a collection of all three-state nets from the design. The tool searches the entire design hierarchy. It returns three-state nets from the entire design, not just from the top level.

If you specify a value for *input_coll*, the command searches for three-state nets only from the specified collection, rather than from the entire design.

Only the nets that are driven by a pin that is tristate or bidirectional are returned as three-state nets.

If you do not specify the **-nets** option, the **all_threestate** command returns an error.

A standard Tcl collection is returned. All of the generic collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, can be performed on this collection handle.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns a collection of all three-state nets from the design:

```
prompt> all_threestate -nets  
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

SEE ALSO

```
all_fanout(2)  
foreach_in_collection(2)  
sizeof_collection(2)
```

all_tieoff_cells

Returns a collection of all tie-off cells in the current design or in the input collection.

SYNTAX

```
collection all_tieoff_cells  
[input_coll]
```

Data Types

input_coll collection

ARGUMENTS

input_coll

Searches the specified input collection and returns a collection of tie-off cells. Objects are to be searched only from the content of the specified input collection, rather than from the design.

DESCRIPTION

The **all_tieoff_cells** command returns a collection of all tie-off cells from the current design. The tool searches the entire design hierarchy. It returns tie-off cells from the entire design, not just from the top level.

Tie-off cells are the constant cells that the tool introduced in the design. To find the value of a tie-off cell, the tool examines the corresponding `lib_cell` of each cell to determine if it is logic-0 or logic-1.

If you specify a value for *input_coll*, the command searches for tie-off cells or nets only from the specified collection, rather than from the entire design.

This command returns a standard Tcl collection. All of the generic collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, can be performed on this collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns a collection of all tie-off cells from the design:

```
prompt> all_tieoff_cells  
{logic1 logic0}
```

The following example returns a collection of all tie-off cells from an existing collection stored in \$COLL:

```
prompt> all_tieoff_cells $COLL  
{logic1}
```

SEE ALSO

`foreach_in_collection(2)`
`sizeof_collection(2)`

all_upf_repeater_cells

Returns a collection of repeater cells available in the design by virtue of the UPF repeater-supply port attribute.

SYNTAX

collection **all_upf_repeater_cells**

ARGUMENTS

None

DESCRIPTION

The **all_upf_repeater_cells** command returns a collection of the repeater cells in the design, that were inserted or matched at the domain boundary pin to honor the specification of the repeater-supply UPF attribute on the port. Repeater cells are buffers and inverters using supplies specified by the **set_port_attributes** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

set_port_attributes(2)

analyze

Analyzes the specified HDL source files and stores the resulting templates into the specified library in a format ready to specialize and elaborate to form linkable cells of a full design.

SYNTAX

status **analyze**

```
[-format verilog | sverilog | vhdl]
[-define define_list]
[-library library_name | -work library_name]
[-uses design_libs_list]
[-vcs vcs_opts]
[-create_update]
[-update]
[-recursive]
[-autoread]
[-rebuild]
[-output_script output_string]
[-exclude exclude_list]
[-verbose]
[-top top_design_name]
file_list
```

Data Types

```
define_list      string
library_name     string
design_libs_list  string
vcs_opts         string
output_string    string
exclude_list     list
top_design_name  string
file_list        list
```

ARGUMENTS

-format verilog | sverilog | vhdl

Specifies the format of the files to be analyzed. The supported formats are Verilog, SystemVerilog, and VHDL.

Specifies that the **-autoread** option should look for files in the specified language whose extensions match the extensions in the list identified by the **hdlin_autoread_verilog_extensions**, **hdlin_autoread_sverilog_extensions**, or **hdlin_autoread_vhdl_extensions** variable. The **-autoread** option supports Verilog, SystemVerilog, and VHDL formats. By default, **-autoread** reads in all the files it finds that have Verilog, SystemVerilog, and VHDL extensions. You can use the -

autoread option to specify file names in the *file_list* that do not have one of the language-specific extensions.

-define *define_list*

Specifies a list of constant macros to be defined for the session of the **analyze** command. This is useful for the inclusion of Verilog or SystemVerilog code enclosed by ``ifdef`` `endif` constructs. For details, see the *HDL Compiler for Verilog User Guide*.

-library *library_name*

Specifies *library_name* as the current work library (whose **-path** receives all analyzed output files). The *library_name* should be previously established by the **define_design_lib** command. Design libraries (directories containing template intermediate files) should not be confused with .db or .ddc linkage libraries (files containing design cells).

By default, the **analyze** command stores all output in the work design library. To store design elements in libraries other than work library, use the **-library** option. A writable directory must exist at the path defined for the current work library.

-work *library_name*

An alias for **-library**. This option is not available with the **-autoread** option.

-uses *design_libs_list*

This option is not available with the **-format vhdl** or **-autoread** option.

SystemVerilog and Verilog designs might contain unresolved references to unelaborated design templates without indication of where those template definitions are located.

The compiler and linker always search for these templates first in the home directory of the parent design. By default, they then continue each template search by visiting every design library defined by the **define_design_lib** command. The **-uses** option overrides this last step. It stores a design library search order (as a cell attribute named **link_design_libraries**) within each design being analyzed. Later, elaboration or linking will follow this prescribed order rather than visiting every known library. An empty list argument limits the search to just the home library of the parent design. You do not need to repeat the **-library_name** setting, and you cannot override a search of the work directory.

-vcs *vcs_opts*

Specifies all VCS-specific command-line options. The options must be specified as a single string, such as enclosed within braces ({}) or double quotation marks (""). Options specified by the **-vcs** option follow the VCS command-line syntax:

[-sverilog | -verilog] Specifies the format of the files to be analyzed.

[-y *directory_path*] Specifies the directories that contain the library files to be searched for unresolved module instantiations in the design.

[+libext+*extension1*+...] Specifies the extension to consider during a file search in the -y library directories. The default is no extension.

[-v *library_file*] Holds several module definitions to be used during the search for unresolved modules.

[-f *command_file*] Specifies a command file.

[+define+*macro_name*+...] Defines a macro.

[+incdir+*dir1*+...] Includes directories in the search list.

src_file1 Specifies the files that are to be analyzed.

This option is not available with the **-autoread** option.

-create_update

Facilitates the packaging of HDL files for distribution performed by developers of DesignWare parts. For details, see the

DesignWare Developer's Guide. This option is not available with the **-autoread** option.

-update

Facilitates the installation of DesignWare parts in DesignWare installation scripts. For details, see the *DesignWare Developer's Guide*. This option is not available with the **-autoread** option.

-recursive

Specifies that the **analyze -autoread** command must recursively scan the directory trees below those specified in the *file_list* list. By default, the **-autoread** option looks only in the specified directories.

This option can be used only with the **-autoread** option.

-autoread

Specifies to use the **-autoread** mode for script-free analysis of whole HDL source directories or directory trees. All HDL source files located in those directories are prescanned, grouped, and then ordered into distinct source code streams based on their mutual inclusion and package-reference dependences. Some or all source streams are finally analyzed using the basic command.

By default, the **-autoread** option creates or replaces only a subset of analyzed result files; it works in an incremental-update mode that resembles the Unix make command. If an HDL source file does not exist or is newer than the intermediate files it should produce, only that file and the source streams that require it are reanalyzed.

For more information, see the description for the **-rebuild** option.

-rebuild

Replaces any result files regardless of their timestamp. It analyzes all the indicated HDL source streams as if the result library was empty.

This option can be used only with the **-autoread** option.

-output_script output_string

Creates a Tcl script that is compatible with Design Compiler and Formality.

The file reading order for third-party tools and a DOT language file dependency graph are embedded in the script as comments.

This option is available only with the **-autoread** option.

-exclude exclude_list

Specifies a list of files and directories that are not to be analyzed. An **analyze -autoread [...] file_list** command checks each HDL source path against all elements of the *exclude_list*; it ignores a file if it or any of its directory paths matches any excluded path.

This option can be used only with the **-autoread** option.

-verbose

Prints out more messages for the **-autoread** option.

This option can be used only with the **-autoread** option.

-top top_design_name

Identifies the top-level component of a hierarchical design specified within the **-autoread** HDL source directories.

The **analyze -autoread -top design_name** command selects a subset of the indicated HDL source streams to analyze. Its goal is to analyze into intermediate form precisely those source files required for a later elaboration and linking of *design_name*. Files in the indicated directories that are not required to elaborate the design hierarchy descending from *design_name* are prescanned

but the design templates they would produce are not created or replaced.

If the **-top** option is not provided, all HDL source streams indicated by the **file_list** argument are analyzed.

This option can be used only with the **-autoread** option.

file_list

Specifies the files to be analyzed.

To analyze multiple files, list them by using curly braces, {}. Note: Source files in a list are effectively concatenated into a single source stream in the listed order.

In **-autoread** mode only, the *file_list* argument can also specify directory names. All HDL source files located in those directories are prescanned. For more information about prescanning, see the descriptions for the **-autoread** and **-recursive** options.

DESCRIPTION

This command translates the specified HDL files and stores the intermediate format in the specified library.

The following paragraphs describe the support for the **-autoread** option:

The **-autoread** option reads in the files from *file_list*, determines the dependencies between them, and analyzes them in the right order to prevent errors due to missed or misplaced files.

The dependencies are calculated only from the files or directories present in *file_list*. If *file_list* changes between consecutive calls with the **-autoread** option, the dependency inference is performed over the latest set of files provided. Based on this, all required sources must be present in *file_list* option or be available if the **-recursive** option is used on each call with the **-autoread** option.

If the top design name is provided by the **-top** option, only the source files required for elaborating that top design are analyzed. This filtering is based on file dependencies previously inferred. If the **-top** option is not provided, all sources are ordered, grouped, and analyzed per the dependency inferred between them.

To locate the source files, the command expands each item in the *file_list* list with the **search_path** variable. Then, it determines whether the result is excluded by the **-exclude** option or the **hdlin_autoread_exclude_extensions** variable. If it is not excluded and a file ends with one of the extensions in the **hdlin_autoread_vhdl_extensions** variable list, it is considered a VHDL source file. If the file ends with one of the extensions in the **hdlin_autoread_verilog_extensions** variable list, it is considered a Verilog source file. If the file ends with one of the extensions in the **hdlin_autoread_sverilog_extensions** variable list, it is considered a SystemVerilog source file.

When expanding a directory, the command collects the files from the directory, and from its subdirectories and theirs recursively if the **-recursive** option is set. The command then performs all extension checks on these files. If the **-format** option is set, only files with Verilog, SystemVerilog, or VHDL extensions are collected based on the value of the option.

After the **-autoread** option collects all of the source files, it performs the following dependency checks:

- Detects analyze dependencies: List RTL files in the right order for analyzing. For example, analyze the file that contains a VHDL entity before analyzing files that defines architectures of that entity, or analyze the file that contains a SystemVerilog package declaration before the files that import that package.
- Detects Verilog and SystemVerilog compilation unit dependencies: Determine if a file needs to be analyzed in the same compilation unit with other files. For example, if a file defines macros, SystemVerilog local parameter, and SystemVerilog enumerated values, and the file is not explicitly included by the file that requires that definitions, the **-autoread** option groups them in the same compilation unit in the correct order. This might not always be possible, such as when a macro is defined several times in different files and the **-autoread** option cannot determine which of those alternatives is the right choice.
- Detects link dependencies: Schedule the analyze stage for files required for elaboration of the design hierarchy. For example,

if a Verilog or SystemVerilog design that is instantiated in one source file is declared on a different source file provided in the *file_list*, the second file also requires to be analyzed for a complete top-down design elaboration.

- Detects include dependencies: If a Verilog or SystemVerilog file is included in another source and the file changes between two consecutive calls of the **-autoread** option (in the same synthesis session), the **analyze** command includes this file and all files that include them to update the design.
- Infers the target library for VHDL files. However, if the **-library** option is specified, this step is skipped and VHDL files provided in the *file_list* are analyzed into the specified design library.

After the dependency check, all the required HDL files specified by the *file_list* option (regarding its dependencies and the **-top design** option if defined) are analyzed.

When the **-autoread** option is used again (with the same *file_list* setting) after a file is changed, only the updated source files are analyzed.

The **-autoread** option executes the dependency analysis based only on the current *file_list* information. All HDL source files that might have relevant dependency relationships should be passed together in one **analyze -autoread** command.

EXAMPLES

The following example analyzes the packages.vhd file and stores the results in the MY_LIB design library:

```
prompt> analyze -work MY_LIB -format vhd packages.vhd
```

The following example analyzes two SystemVerilog clients of two libraries:

```
prompt> define_design_lib BUSDEFN -path /remote/IP/metra/big_bus
prompt> define_design_lib NCLib -path ../NCD/libNC
prompt> analyze -format sverilog -uses { BUSDEFN NCLib } { senderAlice.sv receiverBob.sv }
```

The following example assumes that the current directory is the source directory. It specifies the source file list early in the command line and gives options, including the name of the top-level entity later.

```
prompt> analyze {..} -autoread -recursive -top E1
```

The next example specifies extensions for Verilog files that are different from the default ({.v}), sets the source list and the exclude list, and calls the command with the name of the top-level module name, forcing it to only collect files with Verilog extensions:

```
prompt> set hdlin_autoread_verilog_extensions {.ve .VE}
prompt> set my_sources {mod1/src mod2/src}
prompt> set my_excludes {mod1/src/incl mod2/src/incl}
prompt> analyze $my_sources -exclude $my_excludes -autoread \
-format verilog
```

Note that excluding include directories explicitly is only necessary if include files have the same extensions as source files and not all include files are included in the source.

SEE ALSO

define_design_lib(2)
elaborate(2)


```
link(2)
read_file(2)
report_design_lib(2)
hdlin_autoread_exclude_extensions(3)
hdlin_autoread_verilog_extensions(3)
hdlin_autoread_sverilog_extensions(3)
hdlin_autoread_vhdl_extensions(3)
```

analyze_datapath

Provides a datapath analysis report that lists the resources and datapath blocks used in the current design.

SYNTAX

```
analyze_datapath  
[-file file_name]
```

Data Types

```
file_name  string
```

ARGUMENTS

-file *file_name*

Specifies the file name of a log file that contains the **report_resources -hierarchy** and **report_area -designware** reports from a previously compiled design. This option enables you start the tool and use the **analyze_datapath** command to generate a datapath analysis report for a design without having to read in and compile the design a second time.

DESCRIPTION

The **analyze_datapath** command provides a datapath analysis report that lists a summary of the resources and datapath blocks used in the current design. The report includes information reported by the **report_resources -hierarchy** command and the **report_area -designware** command.

The report displays the area of singleton DesignWare designs and extracted datapath designs. The report summarizes the number of singleton DesignWare components, summarizes the number of datapath designs, and lists the designs output width ranges.

EXAMPLES

The following example displays resource information for the current design:

```
prompt> analyze_datapath
```

```
*****
```

```
Estimate of Datapath Content
```

Extracted Datapath: area = 45.9%, parts = 125
 Singleton Datapath: area = 22.4%, parts = 168

Total Datapath: area = 68.3%, parts = 293

Estimate of Datapath Types

Singletons:

		out	width		
component	count	min	max	ave	
DW01_add	69	7	32	23.5	
DW01_sub	38	9	16	11.5	
DW01_inc	19	9	20	16.4	
DW_cmp	6	12	16	14.8	
DW_mult_uns	35	8	32	19.1	
DW_mult_tc	1	21	21	21.0	

Datapaths:

		out	width		
operation	count	min	max	ave	
+ -	340	6	32	16.8	
*	50	15	31	22.6	
?	69	9	32	15.4	
== !=	5	12	12	12.0	

In the following example, the resource_area_rpt.txt file contains the **report_resources -hierarchy** and **report_area -designware** reports from a previously compiled design. The **-file** option is used to read the resource_area_rpt.txt file and generate a datapath analysis report based on the data in the file. You do not need to read in and compile the design a second time.

prompt> **analyze_datapath -file resource_area_rpt.txt**

Estimate of Datapath Content

Extracted Datapath: area = 45.9%, parts = 125
 Singleton Datapath: area = 22.4%, parts = 168

Total Datapath: area = 68.3%, parts = 293

Estimate of Datapath Types

Singletons:

		out	width		
component	count	min	max	ave	
DW01_add	69	7	32	23.5	

DW01_sub	38	9	16	11.5
DW01_inc	19	9	20	16.4
DW_cmp	6	12	16	14.8
DW_mult_uns	35	8	32	19.1
DW_mult_tc	1	21	21	21.0

Datapaths:

		out width		
operation	count	min	max	ave

+ -	340	6	32	16.8
*	50	15	31	22.6
?	69	9	32	15.4
== !=	5	12	12	12.0

SEE ALSO

report_resources(2)
report_area(2)

analyze_datapath_extraction

Analyzes DesignWare datapath extraction.

SYNTAX

```
status analyze_datapath_extraction  
  [-no_autoungroup]  
  [-no_report]  
  [-filter_msg none | low | medium]  
  [-sort_msg]  
  [-max_msgs max_num_msgs]  
  [-html_file_name html_file]  
  [design_list]
```

ARGUMENTS

-no_autoungroup

Specifies that automatic ungrouping is disabled. All hierarchies are preserved unless otherwise specified. The **-no_autoungroup** option in **analyze_datapath_extraction** behaves the same way as the **-no_autoungroup** option in **compile_ultra**. If you plan to specify the **-no_autoungroup** option in **compile_ultra**, you should also specify the **-no_autoungroup** option in **analyze_datapath_extraction**.

-no_report

Specifies not to report contained resources at the end of the analysis.

-filter_msg none | low | medium

Turns on message filtering. Messages with severity levels less than or equal to the specified level are filtered out.

-sort_msg

Sorts the leakage detection messages (HDL-120, HDL-132) in each design based on their potential impact from high to low.

-max_msgs *max_num_msgs*

Specifies the maximum number of leakage detection messages (HDL-120, HDL-132) to display in each design. The message sorting **-sort_msg** option is turned on automatically.

-html_file_name *html_file*

Writes all outputs to the specified HTML file.

design_list

Specifies list of designs to run datapath extraction analysis.

DESCRIPTION

This command analyzes the datapath extraction of the current design.

When a DesignWare operator is not extracted, it analyzes the reason why it is not extracted. If it is due to the RTL coding, the tool issues a message about the RTL coding style that interferes with datapath extraction.

After the analysis, a report shows the contained resources in the design. This report helps to find the resources in the design source code. Note that the datapath names in this report can differ from the datapath names in the report generated using the **report_resources** command after compilation. However, the names of the contained resources will be the same.

For more information about RTL coding styles that help improve datapath extraction, see to the coding guidelines documented in "Coding Guidelines for Datapath Synthesis" on SolvNet (<https://solvnet.synopsys.com/retrieve/015771.html>).

Here are examples of possible warning messages issued by the **analyze_datapath_extraction** command:

"HDL-120": Issued when a DesignWare operator is breaking the datapath due to leakage on its fanout or the fanout of its driver. This DesignWare operator can be the boundary node of an extracted datapath block.

Datapath leakage occurs when an internal operand is not wide enough to store the result of an operation but the full result is required later. Operands with leakage must be binary and must be at the boundary of a datapath block. Therefore, operands with leakage will break a potentially larger datapath block into smaller datapath blocks, resulting in suboptimal QoR.

Example 1: module test(a, b, c, d, e, f, o1, o2, o3); input [10:0] a, b; input [5:0] c, d, e, f; output [8:0] o1; output [9:0] o2; output [7:0] o3; assign o1 = a + b + c; (add_9, add_9_2) assign o2 = f + o1; (add_10) assign o3 = e + o1; (add_11) endmodule

In this design, the input 'o1' of 'add_10' is truncated. Therefore, the following message is issued:

Information: Operator associated with resources 'add_10' in design 'test' breaks the datapath extraction because there is leakage due to truncation on the fanout of its driver 'add_9 add_9_2'. (HDL-120)

On the other hand, the width of 'o3' is smaller than the width of 'o1'. Therefore, there is no leakage on the input of 'add_11'.

Example 2: module leakage_a (a, b, c, z); input signed [7:0] a, b; input [7:0] c; output [9:0] z; wire signed [8:0] t; assign t = a + b; //add_6 // leakage: signed result used in wider unsigned expression assign z = t + c; endmodule

There is leakage due to sign mismatch at the fanin(signed) and fanout(unsigned) of add_6:

Information: Operator associated with resources 'add_6' in design 'leakage_a' breaks the datapath extraction because there is leakage due to driver/load sign mismatch. (HDL-120)

Example 3: module leakage_b (a, b, c, z); input [7:0] a, b; input signed [7:0] c; output signed [10:0] z; wire signed [8:0] t; assign t = a + b; // add_7 // leakage: unsigned result used in wider signed expression assign z = t + c; endmodule

There is leakage due to sign mismatch at the fanin(unsigned) and fanout(signed) of add_7:

Information: Operator associated with resources 'add_7' in design 'leakage_b' breaks the datapath extraction because there is leakage due to driver/load sign mismatch. (HDL-120)

Subtraction can give negative results that cannot be represented with an unsigned operand of limited width. It is recommended to declare the operands as signed. Note that constants used in signed operations need to be marked signed as well.

Example 4: It is recommended to change the following RTL code: input [13:0] a ; input [13:0] b ; wire [15:0] o = a - 14'd4 - b ;

to

input signed [13:0] a ; input signed [13:0] b ; wire signed [15:0] o = a - 14'sd4 - b ;

The "HDL-120" message is associated with coding guideline 19. (<https://solvnetsynopsys.com/retrieve/015771.html#leakage>).

"HDL-121": There is a sequential cell that breaks the datapath extraction.

Example: module test (clk, en, a, b, c, z); input clk, en; input [7:0] a, b; input [15:0] c; output [15:0] z; reg [15:0] prod;

always @ (posedge clk) begin if (en) begin prod <= a * b; end end assign z = prod + c; endmodule

There is a register between the multiplier and the adder. The following message will be issued:

Information: There is sequential cell in between operator associated with 'mult_11' and 'add_15' in design 'test'. (HDL-121)

The extracted datapath block cannot contain sequential cells (for example, registers). The register in between operators will break a potential larger datapath block into smaller datapath blocks, resulting in suboptimal QoR.

The "HDL-121" message is associated with coding guideline 12. (<https://solvnetsynopsys.com/retrieve/015771.html#pipelining>).

"HDL-125": Design contains instantiated DW. If the instantiated DW is replaced with an inferred DW OP, it could be extracted.

Example: module test(a,b,c,z); parameter wl = 5; input [wl-1:0] a,b,c; output [2*wl-1:0] z;

wire [wl-1:0] sum; assign sum = (b+a); DW02_mult #(wl, wl)U_mult(sum, c, 1'b0, z); endmodule

The instantiated DW02_mult can be replaced by an inferred multiplier. The following message will be issued:

Information: Cell 'U_mult' in design 'test' can not be extracted because it instantiates 'DW02_mult', which could be inferred with operator '*' instead. (HDL-125)

The instantiated DesignWare component cannot be extracted into datapath blocks. If the RTL design uses inferred the DesignWare operation instead of instantiated DesignWare components, the inferred DesignWare operator might be extracted into a datapath block.

The "HDL-125" message is associated with coding guideline 11. (https://solvnetsynopsys.com/retrieve/015771.html#component_instantiation).

EXAMPLES

The following example shows the analysis report about leakage that breaks the datapath extraction.

prompt > **analyze_datapath_extraction**

Information: Checking out the license 'DesignWare'. (SEC-104)

Running analyze_dp_extraction ...

Information: Operator associated with resources 'add_8 add_8_2' in design 'test' breaks the datapath extraction because there is leakage due to truncation on its fanout. (HDL-120)

Report : resources

Design : test

Version: G-2012.06-ALPHA4

Date : Fri Mar 2 14:50:01 2012

Resource Report for this hierarchy in file

/remote/source/rtl/dpl.v

=====

Cell	Module	Parameters	Contained Operations
DP_OP_12J2_124_2007			
	DP_OP_12J2_124_2007		
DP_OP_13J2_125_9654			
	DP_OP_13J2_125_9654		

Datapath Report for DP_OP_12J2_124_2007

Cell	Contained Operations
DP_OP_12J2_124_2007	add_8 add_8_2

Var	Type	Class	Width	Expression
I1	PI	Unsigned	9	
I2	PI	Unsigned	9	
I3	PI	Unsigned	6	
O1	PO	Unsigned	9	I1 + I2 + I3

Datapath Report for DP_OP_13J2_125_9654

Cell	Contained Operations
DP_OP_13J2_125_9654	add_9 add_9_2 add_9_3 add_9_4

Var	Type	Class	Width	Expression
I1	PI	Unsigned	11	
I2	PI	Unsigned	6	
I3	PI	Unsigned	6	
I4	PI	Unsigned	6	
I5	PI	Unsigned	9	
O1	PO	Unsigned	12	I1 + I2 + I3 + I4 + I5

No implementations to report

No multiplexors to report

1

SEE ALSO

report_resources(2)

analyze_dw_power

Reports the DesignWare delay and power contribution.

SYNTAX

```
int analyze_dw_power  
  [-nosplit]  
  [-hierarchy]  
  [-sort slack | dyn_pwr | lkg_pwr]  
  [-sort_ascending]
```

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing tools to extract information from the report output. Most design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-hierarchy

Reports the DesignWare information in the hierarchy of the current design or the current instance. By default, resources used only in the top level of the current design or the current instance are reported.

-sort slack | dyn_pwr | lkg_pwr

Sort the report by the sorting criteria, from largest to smallest.

-sort_ascending

Sort the report in ascending order. This option only works when the **-sort** option is also specified.

DESCRIPTION

This command lists the slack and power contribution of synthetic cells in the current design.

For the generated DesignWare netlist, the command reports the best case slack that the generator can achieve during optimization. For the DesignWare netlist with static implementation, the slack reported is the worst case for the design. If the design is ungrouped, the slack of the ungrouped design is reported. Otherwise, the slack of the design is reported.

For the static DesignWare netlist, only the negative slacks are reported.

This report also shows the power contribution of all the synthetic modules in the current design.

If a synthetic module is ungrouped before propagating the timing and power context, the command does not report the delay and power contribution. For these cells, the **path group** column of the report is empty. The **DW slack** column is **na**. For these DesignWare designs, because they are ungrouped after they are optimized for area or power, setting the **power_effort** to medium or high might not improve power.

The report does not include DesignWare with small bitwidth that are ungrouped. The power of such DesignWare cells are not accounted for the total DesignWare power contribution. Therefore, the total number of DesignWare reported in this report might be different from the total number of DesignWare reported by the **report_area** or the **report_resources** command.

The power contribution of each DesignWare cell is shown in **dyn power %** and **lkg power %** columns. If the power contribution of a DesignWare cell is very small, this column might show a value of **0.00**. The power of these DesignWare cells is included in the total DesignWare power.

The power consumption of all synthetic cells are summarized in the power summary of this report. The slack information is not reported for DesignWare netlists with static implementation. So, the number of cells listed in the **dw_power_analysis** table might not match the number in the power summary report.

For hierarchical DesignWare netlists, only the slack and power contribution of the top-level design are reported. The subcomponents of the DesignWare are not reported.

To enable reporting of the ungrouped DesignWare components, set the **synlib_enable_analyze_dw_power** variable to 1 before compiling the design.

EXAMPLES

The following example generates the **analyze_dw_power** report:

```
prompt> analyze_dw_power
```

```
*****
Report : dw_power_analysis
Design : test
Version: E-2010.12-SP1
Date  : Wed Dec 1 16:54:45 2010
*****
```

DesignWare Power Contribution report

```
=====
|      |      | dyn | lkg |      |      |      |
|      | Path | DW  | power | power |      |      |
| Cell  | group | slack | %    | %    | opt mode | Attr |
=====
| DP_OP_11_124_4242 |      |      |      |      |      |      |
|      | default | -0.16 | 100.00 | 100.00 | area,speed | u    |
=====
```

Attribute:

u - ungrouped

Power of detected synthetic parts

No DW parts to report!

Estimated power of ungrouped synthetic parts

```

-----
DW cell count: 1
Dynamic Power: 3.3256e-01 100.00%
Leakage Power: 1.8464e-03 100.00%
-----

```

```

Total synthetic cell dynamic power: 3.3256e-01 100.00% (estimated)
Total synthetic cell leakage power: 1.8464e-03 100.00% (estimated)

```

```

Note:
Power Units = 1mW
1

```

The following example generates the **analyze_dw_power** report for the hierarchical DesignWare components:

```
prompt> analyze_dw_power -hier
```

```

*****
Report : dw_slack_power
Design : test
Version: E-2010.12-SP1
Date  : Wed Dec 1 17:19:40 2010
*****

```

DesignWare Power Contribution report

```

=====
|      |      | dyn | lkg |      |      |
|      | Path | DW  | power | power |      |
| Cell  | group | slack | %    | %    | opt mode | Attr |
=====
| add_x_938_1 (DW01_add) |      |      |      |      |      |
|      | default | na | 0.32 | 1.34 | area  | u  |
| DP_OP_257J1_124_2676 |      |      |      |      |      |
|      | default | 3.95 | 5.73 | 5.80 | area,speed | u  |
=====

```

```

Attribute:
u - ungrouped

```

Power of detected synthetic parts

```

-----
No DW parts to report!

```

Estimated power of ungrouped synthetic parts

```

-----
DW cell count: 2
Dynamic Power: 5.5592e+00 98.05%
Leakage Power: 1.0676e-03 97.14%
-----

```

```

Total synthetic cell dynamic power: 5.5592e+00 98.05% (estimated)
Total synthetic cell leakage power: 1.0676e-03 97.14% (estimated)

```

```

Note:
Power Units = 1mW
1

```

SEE ALSO

report_area(2)
report_resources(2)
synlib_enable_analyze_dw_power(3)

analyze_minpwr_library

Displays function, drive strength, area and power characteristics of cells from a target library used in datapath generation.

SYNTAX

```
status analyze_minpwr_library  
  [-legend]  
  [library_list]
```

Data Types

library_list list

ARGUMENTS

-legend

Provides a more detailed description of cell types

library_list

Specifies a list of one or more target libraries to be analyzed. If a list is not specified, libraries from the **target_library** variable are used.

DESCRIPTION

This command displays function, drive strength, area and power characteristics of cells of the library that are used in datapath generation. It also checks for cells or combinations of cells that can improve power QoR during datapath generation.

EXAMPLES

The following examples use the demo class.db library. The **analyze_minpwr_library** command prints some general attributes of the library.

Libraries Used:

class (File: ~/class.db)

Library doesn't specify units for: leakage_power

Library units: time=1ns voltage=1V capacitance=0.100000ff dynamic_power=100nW

Note: library doesn't contain any cells fully characterized for power.

This description is followed by three tables.

The first table lists significant cells used in datapath generation.

=====					
For cells selected for datapath generation...					
=====					
	Largest	Best Small	Best Dynamic	Best Leakage	
Cell Type	Drive	Area Drive	Power	Power	
=====					
Not	B4IP	IVI	B4IP	B4IP	
And2	AN2I	AN2I	AN2I	AN2P	
Xor2	EOI	EOI	EOI	EOP	
Xor3	EO3P	EO3P	EO3P	EO3P	
Nand2	ND2P	ND2I	ND2P	ND2P	
Xnor2	ENI	ENI	ENI	ENP	
Xnor3	EN3P	EN3P	EN3P	EN3P	
AOI21	AO6P	AO6	AO6P	AO6P	
Maj23	*NA*	*NA*	*NA*	*NA*	
Maj23N	AO5P	AO5	AO5P	AO5P	
AddAB	*NA*	*NA*	*NA*	*NA*	
AddABC	*NA*	*NA*	*NA*	*NA*	
AddABCD	*NA*	*NA*	*NA*	*NA*	
AddNABC	*NA*	*NA*	*NA*	*NA*	
Benc	*NA*	*NA*	*NA*	*NA*	
Bmux	*NA*	*NA*	*NA*	*NA*	
Mux2	MUX21HP	MUX21H	MUX21HP	MUX21HP	
Mux4	*NA*	*NA*	*NA*	*NA*	
=====					

NA - the library does not contain a cell that implements this function

The second table shows specific attributes of the cells listed in the first table.

=====					
		Dynamic	Leakage		
Library Cell	Cell Area	Cell Drive	Power	Power	
=====					
AN2I	2.00000	0.12539	2.50000	*INV*	
AO2	2.00000	0.73931	5.00000	*INV*	
AO2P	4.00000	0.37418	10.00000	*INV*	
AO4	2.00000	0.73931	5.00000	*INV*	
AO4P	4.00000	0.37418	10.00000	*INV*	
AO6	2.00000	0.73931	3.75000	*INV*	
AO6P	3.00000	0.37418	7.50000	*INV*	
AO7	2.00000	0.73931	3.75000	*INV*	
AO7P	3.00000	0.37418	7.50000	*INV*	
=====					

INV This power value isn't characterized in the library.

The third table lists characteristics for the lowest area and power versions of the library cells. The area, power, and drive-strength

characteristics are checked for potential problems. Informational warning messages are generated first, followed by the cell characterization table.

The cell characterization table contains the following information for each cell type:

Smallest Cell : The smallest cell that is found in the library
 Total # Drives : Number of different drive strengths for this cell
 Min Area # Drives : Number of different drive strengths for the minimum area version of this cell
 Estim Size : Estimated size of the cell based on the estimated grid size
 Actual Size : Cell size from the library data
 Dynamic Power : Dynamic power from the library data
 Min # Trans : Estimated minimum number of transistors required to implement the cell
 Estim # Grids : Estimated number of grids required to implement the cell
 Actual # Grids : Calculated number of grids for the cell based on the area of the smallest inverter in the library

When a potential problem in the cell is detected, an asterisk (*) is added to the value in the characterization table.

The following example flags the potential problems in some library cells.

Information: Xnor3 Cell too large (too many grids). (UISN-89)

Information: Maj23 Cell does not exist. (UISN-86)

Information: Maj23N Not enough total drives (should have at least 4). (UISN-87)

Information: Maj23N No low power cell (should have at least 2 drives for minimum footprint). (UISN-88)

=====
 Cell Characterization for Lowest Area / Power

Estimated grid size = 0.500000

=====

Cell	Smallest Cell	Total # Drives	Min Area # Drives	Estim Size	Actual Size	Dynamic Power	Min # Trans	Estim # Grids	Actual # Grids
Not	IVI	9	4	1.000	1.000	0.0000	2	2	2
And2	AN2I	4	3	2.000	2.000	0.0000	6	4	4
Xor2	EOI	4	2	3.000	3.000	0.0000	10	6	6
Xor3	EO3	4	2	5.500	7.000	0.0000	20	11	14*
Nand2	ND2I	4	2	1.500	1.000	0.0000	4	3	2
Xnor2	ENI	4	2	3.000	3.000	0.0000	10	6	6
Xnor3	EN3	4	2	5.500	7.000	0.0000	20	11	14*
AOI21	AO6	2*	1*	2.000	2.000	0.0000	6	4	4
Maj23	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	0	0	*NA*
Maj23N	AO5	2*	1*	3.000	3.000	0.0000	10	6	6
AddAB	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	0	0	*NA*
AddABC	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	0	0	*NA*
Benc	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	0	0	*NA*
Bmux	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	0	0	*NA*
Mux2	MUX21H	2*	1*	3.500	4.000	0.0000	12	7	8*
Mux4	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	0	0	*NA*

 =====

SEE ALSO

target_library(3)

analyze_mv_design

Analyzes multivoltage design connections.

SYNTAX

```
status analyze_mv_design  
  [-level_shifter | -always_on]  
  [-from_pin from_pin_list]  
  [-to_pin to_pin_list]  
  [-net target_net]  
  [-verbose]
```

Data Types

```
from_pin_list  list  
to_pin_list    list  
target_net     list
```

ARGUMENTS

-level_shifter

Enables level-shifter analysis, which is based on load-to-driver pin arguments.

-always_on

Enables always-on analysis, which is based on the target net argument.

-from_pin *from_pin_list*

Specifies the driver (leaf cell pin or top-level port) for level-shifter analysis.

-to_pin *to_pin_list*

Specifies the loads (leaf cell pins or top-level ports) for level-shifter analysis.

-net *target_net*

Specifies the net for always-on analysis.

-verbose

Generates detailed information in the report.

DESCRIPTION

The **analyze_mv_design** command performs multivoltage analysis and reports design details that can help you understand multivoltage-related issues. You can perform either level-shifter or always-on analysis by using the **-level_shifter** or **-always_on** option.

The level-shifter analysis lists variable settings, power state table information, signal connections, and library availability information. The input for the level-shifter analysis is a pin-to-pin connection, which can be derived easily from the output of the **check_mv_design** command.

The following variable settings are reported in the level-shifter analysis output:

- Level Shifters on clock nets:

This setting is controlled by the **auto_insert_level_shifters_on_clocks** variable. It specifies the list of clock nets considered for automatic level shifter insertion. If the value is "all", every clock net is considered. The default is an empty string.

- Level shifter on top nets:

This setting is controlled by the **compile_no_new_cells_at_top_level** variable. It controls the capability of the tool to insert level-shifter cells at top level. When the variable value is set to **true**, no new level-shifter cells are accepted. The default is **false**.

- Level shifters allowed on leaf pin boundary:

This setting is controlled by the **mv_allow_ls_on_leaf_pin_boundary** variable. When the setting is **true**, the tool is allowed to insert level shifters at any driver or load pins of a net, even if the pins are not on a power domain boundary. The default is **false**.

The always-on analysis reports details about always-on constraints, related supply nets, relevant power state table settings, and library cell availability for the target net.

EXAMPLES

The following command analyzes a specified level-shifter connection:

```
prompt> analyze_mv_design -level_shifter \
    -from_pin gprs_nrestore -to_pin GPRs/A_reg_reg[0]/NRESTORE
...

Level Shifter Analysis (1/1)
from driver pin 'gprs_nrestore', to load pin 'GPRs/A_reg_reg[31]/NRESTORE'

*****
Level Shifter Analysis - Settings
*****
Level Shifters on clock nets:      all
Level Shifters on top nets:       TRUE
Level Shifters allowed leaf pin boundary:  FALSE

*****
Level Shifter Analysis - Driver and Load Related PST
*****
Current Scope: ChipTop
```

Supply Names: VDD, VSS, VDDG,
 Resulting Power States in this Scope:
 VDD| VSS|VDDG|
 drv: HV| GND| HV|
 drv: HV| GND| LV|

 Level Shifter Analysis - Drive to Load Pins and Nets

From Pin: gprs_nrestore
 Related Supplies: VDD; VSS;
 To Pin: GPRs/A_reg_reg[31]/NRESTORE
 Related Supplies: VDDG; VSS;

Pin: =domain boundary= gprs_nrestore;
 Type: Design Port; Domain Boundary;
 Constraint: not defined

Net: gprs_nrestore;
 Domain: TOP;
 Local Fanout: 1;
 Primary Supply Nets: VDD, VSS,
 Available Supply Nets: VDD, VSS, VDDI, VDDIS, VDDG, VDDM,
 - Driver pin related supply nets (VDD, VSS) and load pin related supply nets
 (VDDG, VSS) are available in the domain 'TOP'
 - Driver pin related supply nets (VDD, VSS) are the primary supplies in the
 domain 'TOP'

Pin: =domain boundary= GPRs/gprs_ret1;
 Type: Hierarchical Port; Domain Boundary;
 Constraint: not defined

Net: GPRs/gprs_ret1;
 Domain: GPRS;
 Local Fanout: 608;
 Primary Supply Nets: VDDGS, VSS,
 Available Supply Nets: VSS, VDDG, VDDGS,
 - Neither driver pin related supply nets (VDD, VSS) nor load pin related
 supply nets (VDDG, VSS) are primary supplies in the domain 'GPRS'

Pin: GPRs/A_reg_reg[31]/NRESTORE;
 Type: Cell Input Pin; Lib Cell: RSDFCSRHD2BWP;

 Level Shifter Analysis - Target Libraries

Required pin-to-pin level shifter type: H2L
 All level shifters cells found in the target libraries: 12
 Usable level shifter cells: 12
 Usable level shifter cells of type H2L: 4
 Lib Cell 'mylib_wc09072/LVHL' works in the required voltage range:
 Input voltage range [0.700 - 1.300]
 Output voltage range [0.700 - 1.300]
 Lib Cell 'mylib_wc0909/LVHL' works in the required voltage range:
 Input voltage range [0.700 - 1.300]

Output voltage range [0.700 - 1.300]
Lib Cell 'mylib_bc11088/LVHL' works in the required voltage range:
Input voltage range [0.700 - 1.300]
Output voltage range [0.700 - 1.300]
Lib Cell 'mylib_bc1111/LVHL' works in the required voltage range:
Input voltage range [0.700 - 1.300]
Output voltage range [0.700 - 1.300]

1

SEE ALSO

check_mv_design(2)
report_mv_library_cells(2)

analyze_rtl_congestion

Reports pre-synthesis congestion analysis.

SYNTAX

status **analyze_rtl_congestion**
[-nosplit]

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing application to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **analyze_rtl_congestion** command reports pre-synthesis congestion analysis for the current design. The command evaluates the RTL for potential congestion issues and reports the issues so you can resolve them early in the design flow.

The report includes the line numbers in the RTL where the issues occur to help you identify where RTL changes are needed.

The following types of structures are known to cause congestion:

- Large MUXes
- Large data switches
- Large selectors
- Large ROMS
- Parallel high-fanout nets

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **analyze_rtl_congestion** command:

```
prompt> analyze_rtl_congestion

*****

Report : RTL congestion
Design : mem
Version: I-2013.12-SP1-CS1
Date  : Thu Dec 12 03:54:00 2013
*****

#####
Large MUX details
#####

-----
Structure  Cell name      Size (width)  Line No  RTL File
-----
MUX_OP     C10567      16 x 1 (163)   83  /u/9000560869.v
MUX_OP     C10566      16 x 1 (163)   82  /u/9000560869.v

#####
Large Data Switch details
#####

-----
Structure  Cell name      Size (width)  Line No  RTL File
-----
DATA_SWITCH
    dataline_A_0/C76676 2 x 1 (128)   361  /rtl/dataline.v
    dataline_A_0/C76672 2 x 1 (128)   365  /rtl/dataline.v
DATA_SWITCH
    dataline_A_0/C76614 2 x 1 (8)     159  /rtl/dataline.v
    dataline_A_0/C76613 3 x 1 (8)     159  /rtl/dataline.v

#####
Large ROM details
#####

-----
Structure  Cell name      Size (width)  Line No  RTL File
-----
ROM        C81944      4096 x 1 (21)   9   /rtl/large_rom2.v

#####
Large Selector details
#####

-----
Structure  Cell name      Size (width)  Line No  RTL File
-----
```

```
SELECT_OP C117247      151 x 1 (4)   83634 /rtl/pba_csr32.v
SELECT_OP C117242      190 x 1 (1)   83634 /rtl/pba_csr32.v
```

```
#####
Parallel High Fanout Net details
#####
```

Structure	Net name	Cell name	Loads	Line No	RTL File
PARALLEL_HFN					
	sel[0][0][0]	sel_reg[0][0][0]	128	18	/rtl/reg-file-2.sv
	sel[0][0][1]	sel_reg[0][0][1]	128	18	/rtl/reg-file-2.sv
PARALLEL_HFN					
	sel[0][1][0]	sel_reg[0][1][0]	128	18	/rtl/reg-file-2.sv
	sel[0][1][1]	sel_reg[0][1][1]	128	18	/rtl/reg-file-2.sv

1

SEE ALSO

report_congestion(2)

append_to_collection

Adds objects to a collection and modifies a variable.

SYNTAX

```
collection append_to_collection
[-unique]
  var_name
  object_spec
```

Data Types

<i>var_name</i>	collection
<i>object_spec</i>	list

ARGUMENTS

-unique

Indicates that duplicate objects are to be removed from the resulting collection. By default, duplicate objects are not removed.

var_name

Specifies a variable name. The objects matching *object_spec* are added into the collection referenced by this variable.

object_spec

Specifies a list of named objects or collections to add.

DESCRIPTION

The **append_to_collection** command allows you to add elements to a collection. This command treats the variable name given by the *var_name* option as a collection, and it appends all of the elements in *object_spec* to that collection. If the variable does not exist, it is created as a collection with elements from the *object_spec* as its value. If the variable does exist and it does not contain a collection, it is an error.

The result of the command is the collection that was initially referenced by the *var_name* option, or the collection created if the variable did not exist.

The **append_to_collection** command provides the same semantics as the common use of the **add_to_collection** command; however, this command shows significant improvements in performance.

An example of replacing the **add_to_collection** command with the **append_to_collection** command is provided below. For example,

```
set var_name [add_to_collection $var_name $objs]
```

Using the **append_to_collection** command, the command becomes:

```
append_to_collection var_name $objs
```

The **append_to_collection** command can be much more efficient than the **add_to_collection** command if you are building up a collection in a loop. The arguments of the command have the same restrictions as the **add_to_collection** command. For more information about these restrictions, see the **add_to_collection** man page.

EXAMPLES

The following example from PrimeTime shows how a collection can be built up using the **append_to_collection** command:

```
prompt> set xports
Error: can't read "xports": no such variable
      Use error_info for more info. (CMD-013)
prompt> append_to_collection xports [get_ports in*]
{in0 in1 in2}
prompt> append_to_collection xports CLOCK
{in0 in1 in2 CLOCK}
```

SEE ALSO

- `add_to_collection(2)`
- `foreach_in_collection(2)`
- `index_collection(2)`
- `remove_from_collection(2)`
- `sizeof_collection(2)`

apply_clock_gate_latency

Annotates the clock latencies on the existing clock-gating cells based on the settings previously specified using the **set_clock_gate_latency** command.

SYNTAX

status **apply_clock_gate_latency**

ARGUMENTS

The **apply_clock_gate_latency** command has no arguments.

DESCRIPTION

The **apply_clock_gate_latency** command annotates the clock latencies on the existing clock-gating cells based on the settings previously specified using the **set_clock_gate_latency** command.

Use this command to do the following:

- Manually annotate clock-latency values specified by the **set_clock_gate_latency** command to existing clock-gating cells
- Modify clock-latency values applied to clock-gating cells after gate-level clock gating is finished.

If an inconsistent clock-latency setting is detected during annotation, the tool tries to resolve it to produce a situation where clock latencies are monotonically increasing values in the path going from the clock source to the gated registers. The following actions can be performed to achieve this:

- Assume that clock-gating cell A is directly driving one or more registers, and some clock-latency settings have been provided for the timing of these gated registers by using value 0 for the **stage** option of **set_clock_gate_latency** command or by propagation of latency specified for the clock object. In this situation, the latency annotated on cell A is higher than the latency provided for gated registers, then the latency on clock-gating cell A is overwritten with the latency provided for gated registers. If this fix is done, it is informed by warning message PWR-746.
- If clock-gating cell A is driving another clock-gating cell B, and the clock latency set on A is higher than the one set on B, the tool resolves it by overwriting the clock latency on A with the value set on B. If this fix is done, it is also informed by warning message PWR-746.

To remove the settings specified using **set_clock_gate_latency**, use the **reset_clock_gate_latency** command.

If the variable **power_cg_physically_aware_cg** is enabled, then the annotation of latency values from **set_clock_gate_latency** is disabled.

Multicorner-Multimode Support

The actual clock latency annotation performed during the execution of **compile_ultra** or **apply_clock_gate_latency** commands is done for all the scenarios for which a clock latency value is available.

SEE ALSO

- `power_cg_physically_aware_cg(3)`
- `remove_clock_latency(2)`
- `report_clock_gating(2)`
- `reset_clock_gate_latency(2)`
- `set_clock_gate_latency(2)`
- `set_clock_latency(2)`

apply_power_model

Apply a power model which describes the power intent of a reference library cell to instances.

SYNTAX

```
status apply_power_model
  power_model_name
  [-elements instance_names]
  [-supply_map supply_map_list]
  [-port_map port_map_list]
  [-parameters param_map_list]
  [-all_hard_macros]
```

Data Types

```
power_model_name  string
instance_names    list of instances
supply_map_list   list of power_model_supply_set current_scope_supply_set
port_map_list     list of power_model_supply/logic_port current_scope_supply/logic_net
param_map_list    list of power_model_parameter parameter_value
```

ARGUMENTS

power_model_name

Specifies the name of the power model to be applied. The name should be a simple (non-hierarchical) name.

The applied power model is searched in this descending order of preference:

1. Power models defined in the current scope.
2. Power models defined in any direct ancestor scope: the closest the better.
3. Power models defined in user power model library. Please see variables `upf_power_model_search_path` and `upf_power_model_library`.
4. Power models defined in any scope.

-elements *instance_names*

Specifies a collection of cells that this power model should be applied to.

If this option is not specified, the power model will be applied to all cells in and under the current scope whose reference library cells match those specified in the power model (i.e. library cells listed in -for option of the `define_power_model` command).

-supply_map *supply_map_list*

Specifies the mapping of supply sets in the power model to supply sets in the current scope. For example: `-supply_map {{model_vdd top_vdd} {model_vss top_vss}}`

-port_map *port_map_list*

Specifies the mapping of supply/logic ports in the power model to supply/logic nets in the current scope. For example: `-port_map {{model_supply_port top_supply_net} {model_logic_port top_logic_net}}`

-parameters *param_map_list*

Specifies the mapping of parameters in the power model to actual parameter values. For example: `-parameters {{model_param1 "value_1"} {model_param2 "value_2"}}`

-all_hard_macros

Apply power models to all hard macro cells under the current scope with matching power model, i.e., the macro lib cell is included in the `-for` option of the `define_power_model` command, or the power model name is the lib cell name.

`power_model_name` must not be provided and no other options can be specified.

DESCRIPTION

The `apply_power_model` command binds the `power_model_name` to the cell instances specified by the `-elements` option. The associated parameter binding specified via the `-parameters` option allows objects within the environment to be bound to parameters defined within the power model. Not all parameters within the power model need to be bound to objects in the environment and parameters that are not bound shall retain their default values during run time.

The `apply_power_model` command sets the scope to each of the specified set of instances, referred to as the instance scope, and then executes the set of UPF commands in the power model `power_model_name`. Upon return, the current scope is restored to the scope in which the command was invoked.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example applies a power model to two macro cells:

```
prompt> define_power_model MODEL -for {macro_lib_cell} {
  create_supply_set SS_macro
  create_supply_set SS_macro_ao
  create_supply_port VDD_macro
  create_supply_port VSS_macro
}
prompt> create_supply_set SS_top
prompt> create_supply_set SS_top_ao
prompt> create_supply_net VDD_top
prompt> create_supply_net VSS_top
prompt> apply_power_model MODEL -elements {u1/macro u2/macro}
  -supply_map {{SS_macro SS_top} {SS_macro_ao SS_top_ao}}
```

```
-port_map {{VDD_macro VDD_top} {VSS_macro VSS_top}}
```

The following example applies a power model to all instanced cells of lib_cell in or under the current scope. Power domain PD_macro will be created in each of the macro cells:

```
prompt> define_power_model MODEL -for {lib_cell} {  
  add_parameter DOMAIN -default domain -description "top power domain in model"  
  create_supply_net VDD  
  create_supply_net VSS  
  create_supply_set SS -function {power VDD} -function {ground VSS}  
  create_power_domain PD_${DOMAIN} -supply {primary SS} -include_scope  
}  
prompt> apply_power_model MODEL -parameters {{DOMAIN macro}}
```

The following example applies power model MODEL_A to all instantiated cells of lib_cell_A and applies power model lib_cell_B to all instantiated cells of lib_cell_B in or under the current scope:

```
prompt> sh cat ./power_model.upf  
define_power_model MODEL_A -for {lib_cell_A} { create_power_domain PD_A }  
prompt> sh cat ./lib_cell_B.upf  
define_power_model lib_cell_B { create_power_domain PD_B }  
prompt> set upf_power_model_search_path "."  
prompt> set upf_power_model_library "power_model.upf lib_cell_B.upf"  
prompt> apply_power_model -all_hard_macros
```

SEE ALSO

```
define_power_model(2)  
add_parameter(2)  
report_power_model(2)  
upf_power_model_library(3)  
upf_power_model_search_path(3)
```

apropos

Searches the command database for a pattern.

SYNTAX

```
string apropos
      [-symbols_only]
      pattern
```

Data Types

pattern string

ARGUMENTS

-symbols_only

Searches only command and option names.

pattern

Searches for the specified *pattern*.

DESCRIPTION

The **apropos** command searches the command and option database for all commands that contain the specified *pattern*. The *pattern* argument can include the wildcard characters asterisk (*) and question mark (?). The search is case-sensitive. For each command that matches the search criteria, the command help is printed as though **help -verbose** was used with the command.

Whereas **help** looks only at command names, **apropos** looks at command names, the command one-line description, option names, and option value-help strings. The search can be restricted to only command and option names with the **-symbols_only** option.

When searching for dash options, do not include the leading dash. Search only for the name.

EXAMPLES

In the following example, assume that the **get_cells** and **get_designs** commands have the **-exact** option. Note that without the **-symbols_only** option, the first search picks up commands which have the string "exact" in the one-line description.

prompt> **apropos exact**

get_cells # Create a collection of cells
[-exact] (Wildcards are considered as plain characters)
patterns (Match cell names against patterns)

get_designs # Create a collection of designs
[-exact] (Wildcards are considered as plain characters)
patterns (Match design names against patterns)

real_time # Return the exact time of day

prompt> **apropos exact -symbols_only**

get_cells # Create a collection of cells
[-exact] (Wildcards are considered as plain characters)
patterns (Match cell names against patterns)

get_designs # Create a collection of designs
[-exact] (Wildcards are considered as plain characters)
patterns (Match design names against patterns)

SEE ALSO

help(2)

as_collection

Find a collection by name

SYNTAX

```
string as_collection  
[-check]  
collection1
```

Data Types

```
collection1  string
```

ARGUMENTS

-check

Return 1 if *collection1* is a collection

collection1

The collection to convert

DESCRIPTION

This command looks up a string value and returns it as a collection if the collection is still live. All other values are passed through this command unmodified.

This command also provides a simple way to check if the input is a collection.

EXAMPLES

Check if a value is a collection:

```
prompt> as_collection -check [get_cells]  
1  
prompt> as_collection -check hello  
0
```

SEE ALSO

`collections(2)`

`foreach_in_collection(2)`

associate_supply_set

Associates a supply set handle to another supply set handle or supply set reference.

SYNTAX

```
status associate_supply_set  
    supply_set_name  
    -handle supply_set_handle
```

Data Types

```
supply_set_name    string  
supply_set_handle string
```

ARGUMENTS

supply_set_name

Specifies the name of the supply set to which the supply set handle is associated. The name can be the name of a supply set or a supply set handle.

This is a required argument.

-handle *supply_set_handle*

Specifies the supply set handle on which the action is performed.

This is a required argument.

DESCRIPTION

The **associate_supply_set** command associates the supply set handle specified with the **-handle** option to the specified supply set. After the association, the tool treats corresponding functions in each supply set to be virtually connected. So, the supply sets inherit common switching behavior and must eventually be resolved to the same physical supply net.

A supply set supports the following functions:

- Power
- Ground

Multicorner-Multimode Support

This command has no dependency on scenario specific information.

EXAMPLES

The following example associates primary supply handle of domain PD1 to a supply set *SS1*

```
prompt> associate_supply_set SS1 \  
       -handle PD1.primary  
1
```

SEE ALSO

`create_power_domain(2)`
`create_supply_set(2)`

balance_buffer

Builds a balanced buffer tree on user-specified nets and drivers.

SYNTAX

```
status balance_buffer
  [-from start_point_list]
  [-to end_point_list]
  [-net net_list]
  [-force]
  [-library library_name]
  [-prefer buffer | inverter | lib_cell_name]
```

Data Types

```
start_point_list  string
end_point_list    string
net_list          string
```

ARGUMENTS

-from *start_point_list*

Specifies a list of driver pins from which buffer trees are to be created.

-to *end_point_list*

Specifies a list of loads to buffer.

-net *net_list*

Specifies a list of nets to buffer.

-force

Forces **balance_buffer** implementation regardless of cost effect.

-library *library_name*

Specifies the library where the *lib_cell_name* is contained. You must specify this option when specifying *lib_cell_name*. If you do not specify *lib_cell_name*, this option is ignored.

-prefer buffer | inverter | *lib_cell_name*

Specifies the preferred cell type for the buffer tree.

- When *buffer* is specified, the tool creates a buffer tree using buffer-type library cells.

- When *inverter* is specified, the tool creates a buffer tree using inverter-type library cells.
- When *lib_cell_name* is specified, that library cell is the only cell used to form the buffer tree. You must specify *library_name* when specifying *lib_cell_name*. The library cell must be a buffer or an inverter library cell.

DESCRIPTION

The **balance_buffer** command creates DRC-free balanced buffer trees on user-specified nets and drivers. It removes existing buffer trees on the specified nets or drivers if any are present and builds a new DRC-free buffer tree. The buffer trees created in one hierarchy considers loads and drivers across hierarchies and creates buffer trees at the fanout nets into these hierarchies.

The **balance_buffer** command generates a layered buffer tree, where each stage uses the same buffer or inverter and each gate of a given stage roughly drives the same load capacitance. The type of cells used are a combination of buffer and inverter cells available from the target library, unless the **-prefer** option is specified.

Any input port driving a net that is to be buffered by **balance_buffer** must have its drive value set. Otherwise, the default value specifies that the input port has infinite drive, and no buffer tree is created. The **balance_buffer** command uses the NLDM information specified in the target library for calculations on a buffer tree.

Note that **balance_buffer** does not take clock skew into account and, therefore, is not recommended for clock trees.

EXAMPLES

In the following example, a buffer tree is first built from port i0. Then, two additional buffer trees are built to drive load1 and load2.

```
prompt> read -f edif test.edif
prompt> set_driving_cell -cell IV i0
prompt> balance_buffer -from i0
prompt> balance_buffer -to {load1, load2}
```

SEE ALSO

compile(2)

balance_registers

Moves the registers of the current design to achieve a minimum cycle time.

SYNTAX

status **balance_registers**

DESCRIPTION

The **balance_registers** command moves the registers of a design to obtain a minimum cycle time. You do not have to set the **balance_registers** attribute to invoke **balance_registers** outside of the **compile** command. Using **balance_registers** outside of **compile** usually gives better control and better quality of results because the gate-level netlist used as a starting point is defined exactly by the initial compile.

The **balance_registers** command has the following requirements:

- All registers must be edge-triggered flip-flops clocked by the same phase of the same clock; or, all registers must be master-slave elements with the same signals for the master and slave pins.
- Flip-flops and master-slave elements cannot be present in the same design.
- Master and slave clock waveforms cannot overlap.
- The clock pins of all flip-flops in the design must be connected to the same clock port. The interconnection from the clock port to the clock pins can contain buffer and inverter cells having one input and one output pin. All paths from the clock port to the clock pins of the registers can contain either an even number (including zero) of inverters or an odd number of inverters. These paths cannot be comprised of both even and odd numbers of inverters in a single design.

The outputs of the clock network can be connected to only sequential cells. The clock network of the retimed design will not contain any cells.

- Flip-flops that have asynchronous set or clear pins are not moved.
- Designs cannot contain a combinational loop.
- Set the timing constraints for the design in the following way:
 - The external clock port of the design must have a clock constraint created by the **create_clock** command. This will be called the base clock.
 - All primary inputs of the design must have a non-negative input delay relative to the base clock.
 - All primary outputs of the design must have a non-negative output delay relative to the base clock.
 - Negative input and output delays are tolerated, but the quality of the final retiming result may be worse.

- Input and output delays relative to virtual clocks or clocks other than the base clock are ignored.
- Point-to-point timing exceptions as created by the **set_false_path**, **set_multicycle_path**, **set_max_delay**, and **set_min_delay** commands are tolerated but ignored.
- Case analysis constraints are ignored.
- Designs cannot contain tristate cells or unmapped synthetic library components.

By default, **balance_registers** ungroups all instances in a design before moving registers. Avoid ungrouping by setting the **dont_touch** attribute on instances that are not to be ungrouped. The **balance_registers** command does not ungroup an instance that has the **dont_touch** attribute, and does not place any registers inside an instance that has the **dont_touch** attribute.

The **balance_registers** command includes a delay modeling capability. For predictability, the algorithm selects a flip-flop from the target library with small setup time, good drive at the outputs, and low input loading as compared with other registers in the library. This register is designated as the preferred flip-flop. Sequential mapping is invoked in the final step to improve the circuit by remapping the registers. The preferred flip-flop helps provide tighter bounds on the performance variation after the **balance_registers** sequence. To disable delay modeling, set the shell variable **balance_reg_delay** to 0.

As part of the delay modeling capability, **balance_registers** provides some statistics on the delays in the circuit. If the preferred flip-flop appears as a driver for a net, **balance_registers** analyzes the circuit to compute the clock-pin-to-next-state-pin delay for that net. The maximum of all these clock-pin-to-next-state pin delays is used as a reference for checking the input delays of the design. They should be larger than the average clock-pin-to-next-state pin delay.

The clock-pin-to-next-pin-delay, the setup time of the preferred flip-flop and the clock uncertainty sum up to the clock correction. The sum of the clock correction and the maximum critical path length reported when balance registers has finished the register moving phase give an estimate for the clock period after retiming. However, the value of the clock correction does not influence the minimum clock period achieved. An example detailing the output of the delay modeling capability is shown in the EXAMPLES section. More information on the **balance_registers** command can be found in the Design Compiler documentation.

EXAMPLES

The following example reads in the design pipe_mult_32.db, and moves the registers to achieve a minimum cycle time:

```
prompt> read pipe_mult_32.db
```

```
prompt> balance_registers
```

The output provided by the delay modeling capability shows that the sequential cell F611 from the target library is selected as the preferred flip-flop F611. The relevant delay information of the design is summarized in the two tables titled "Design Analysis" and "Clock-to-Q delay distribution."

```
prompt> balance_registers pipe_test
```

```
Loading design 'pipe_test'
```

```
Beginning retiming
```

```
-----
```

```
Retiming pipe_test
```

```
Preferred register is F611 with setup = 1.34
```

```
Design Analysis
```

```
-----
| Worst | Best | Average |
-----
clock to Q | 1.85 | 1.40 | 1.54 |
-----
```

Clock to Q delay distribution

delay range	number of nets
[1.40 - 1.49]	23
[1.49 - 1.58]	33
[1.58 - 1.67]	14
[1.67 - 1.76]	4
[1.76 - 1.85]	4

Lower bound estimate = 5.40

Critical path length = 5.40

Clock correction = 3.19 (clock-to-Q delay = 1.85, setup = 1.34, uncertainty = 0.

00) Structuring 'pipe_test'

Mapping 'pipe_test'

Retiming complete

Transferring Design 'pipe_test' to database 'mult4_g.db'

1

SEE ALSO

current_design(2)

set_balance_registers(2)

set_dont_touch(2)

attributes(3)

break

Immediately exits a loop structure.

SYNTAX

int **break**

ARGUMENTS

None

DESCRIPTION

The **break** command immediately exits the innermost loop structure. Use the **break** command to terminate a **while** loop, rather than waiting until the loop evaluates to zero.

The **break** command returns the integer 1, indicating successful operation. A syntax error is reported if **break** is used outside a loop structure.

EXAMPLES

In the following example, a list of file names is scanned until the first file name that is a directory is encountered. The **break** command is used to terminate the foreach loop when the first directory name is encountered.

```
foreach f [which {VDD.ave GND.tech p4mvn2mb.idm}] {  
  echo -n "File $f is "  
  if { [file isdirectory $f] == 0 } {  
    echo "NOT a directory"  
  } else {  
    echo "a directory"  
    break  
  }  
}
```

SEE ALSO

`continue(2)`
`while(2)`

cd

Changes the current directory.

SYNTAX

```
status cd  
      [directory]
```

Data Types

directory string

ARGUMENTS

directory

Specifies an existing directory name.

DESCRIPTION

The **cd** command changes the current directory to the specified *directory*. If you do not specify *directory*, your login or home directory becomes the new current directory. By default, the current directory is the directory where the shell is invoked.

A file specification of dot (.) is shorthand for the current directory. Dot is commonly included in the **search_path** variable. Changing the current directory changes the interpretation of "." in the **search_path** variable.

Note that the **sh** command cannot be used to change directories.

EXAMPLES

The following are examples of using the **cd** command:

```
prompt> cd /usr/designer  
prompt> pwd  
/usr/designer
```

```
prompt> cd joe  
prompt> pwd
```

```
/usr/designer/joe

prompt> cd ../bob
prompt> pwd
/usr/designer/bob

prompt> cd ~
prompt> pwd
/usr/designer/joe

prompt> cd ~doug
prompt> pwd
/usr/designer/doug

prompt> cd
prompt> pwd
/usr/designer/joe

prompt> cd /usr/designer
prompt> ls
bob    designs    doug
joe    one.ddc    two.ddc

prompt> cd designs
prompt> ls
one.ddc    other.ddc

prompt> set_app_var search_path ". ~"

prompt> read_ddc {one.ddc two.ddc}
Loading ddc file '/usr/designer/designs/one.ddc'
Loading ddc file '/usr/designer/joe/two.ddc'

prompt> cd /tmp

prompt> read_ddc one.ddc
Loading ddc file '/usr/designer/joe/one.ddc'
```

SEE ALSO

ls(2)
pwd(2)

cell_of

Returns the cell objects for the specified pins in the current design.

SYNTAX

```
list cell_of  
    [object_list]
```

Data Types

object_list list

ARGUMENTS

object_list

Specifies a list of pins in the current design whose cells are returned by the **cell_of** command.

DESCRIPTION

The **cell_of** command returns a list of cell objects that own the pins specified in the *object_list*. If no cells are found, the return value is an empty list ({}). Double quotation marks (") are needed for pin names that are ambiguous to the command parser.

EXAMPLES

The following example returns the cell of the specified pin in the current design:

```
prompt> cell_of U1/P  
{U1}
```

The following example returns the cell of pin "reg(8)/Q" in the current design:

```
prompt> cell_of "reg(8)/Q"  
{reg(8)}
```

The following example returns the list of cells for the given list of pins. The first cell is the owner cell of the first pin and the second cell is the owner cell of the second pin.

```
prompt> cell_of {U3/B U9/A}  
{U3 U9}
```

The following example returns the cell of the given pin. The cell name is a hierarchical name at the current level, which can be found after ungrouping.

```
prompt> cell_of {U3/U1/B}  
{U3/U1}
```

The following example returns a warning message when the pin does not exist in the current design:

```
prompt> cell_of U999/Z  
Warning: Can't find pin `U999/Z' in design `TOP'. (UID-95)  
{}
```

The following example returns the cell of a hierarchical pin:

```
prompt> cell_of U1/U2/U3/A  
{U1/U2/U3}
```

SEE ALSO

`all_connected(2)`

change_link

Changes the design to which a cell is linked.

SYNTAX

```
status change_link
      object_list
      design_name
      [-all_instances]
      [-force]
      [-pin_map pin_map_list]
```

Data Types

```
object_list    list
design_name    string
pin_map_list  list
```

ARGUMENTS

object_list

Specifies the cells or references in the current design for which to change the link.

design_name

Specifies the name of the design to which to link the cells or references in the object list.

-all_instances

Enables the command to accept instance cells in the object list.

-force

Enables the command to allow mismatched pin counts, as long as any mismatched cells in the object list have fewer pins than the specified design.

-pin_map *pin_map_list*

Specifies the pin mapping used to map old pin names to new pin names. The pin name must be the reference cell pin name.

To specify the pin mapping, use the following syntax:

```
{{old_pin1 new_pin1} ... {old_pin_n new_pin_n}}
```

New pins maintain the same net connections as the corresponding old pins.

DESCRIPTION

The **change_link** command specifies a design for which to change the link for a cell or reference.

If you specify a cell in the object list, the command changes it to one occurrence of the specified design. If you specify a reference in the object list, the command changes all cells of the specified reference type to occurrences of the design.

You can change the cell or reference link only to a compatible design. For example, the design must have the same number of ports with the same name and direction as the cell or reference.

Although the *design_name* argument accepts names in the format *library/library_cell*, it does not imply that the actual library cell used for the new cell will be from the specified library. The actual library cell used is determined by the current link library settings.

During **change_link** activity, all Synopsys database format link information is copied from the old design to the new design. If the old design is a synthetic module, all attributes of the old synthetic module are moved to the new link. After running the **change_link** command, link the design with the **link** command.

Use the **-all_instances** option when any cell in the *object_list* argument is an instance in the lower hierarchy and its parent cell is not unique. All similar instance cells under the same parent design are automatically linked to the new reference design. You do not have to change the current design to change the link for the instance cells in the lower design hierarchy. You do not need to use the **-all_instances** option if none of the cells in the *object_list* argument is an instance in the lower hierarchy or its parent cell is unique.

A common use of the **change_link** command for hierarchical cells is to rename designs and to change the cell links to the designs of the new names. In these cases, use the **rename_design** command with the **-update_links** option instead of the **copy_design**, **rename_design**, **change_link** and **remove_design** command set. The **rename_design** command provides faster runtime. For more information, see the **rename_design** command man page.

EXAMPLES

The following example shows a common use of the **change_link** command. The *ADDER* design is copied to a new design named *MY_ADDER*. The *U1* and *U2* cells are then linked from the current design to *MY_ADDER*.

```
prompt> copy_design ADDER MY_ADDER
Copying design 'ADDER' to 'MY_ADDER'
```

```
prompt> change_link {U1 U2} MY_ADDER
```

The following example shows an improper use of the **change_link** command. The specified cell does not have the same number of ports as the design to which it is being linked.

```
prompt> change_link U3 HALF_ADDER
Error: Number of pins on cell 'U3' don't equal design 'HALF_ADDER'
```

The following example uses the **-all_instances** option:

```
prompt> current_design
Current design is 'top'.
{top}

prompt> get_cells -hierarchical -filter "ref_name == bot"
{mid1/bot1 mid1/bot2}
1
```



```
prompt> get_cells -hierarchical -filter "ref_name == IVA"
{mid1/bot1/inv1 mid1/bot2/inv1 mid1/bot1/inv2 mid1/bot2/inv2 mid1/inv3}
1

prompt> change_link mid1/bot1/inv1 lsi_10k/IVAP -all_instances
Information: Changed link for all instances of cell 'inv1' in subdesign 'bot'. (UID-193)
1

prompt> get_cells -hierarchical -filter "ref_name == IVAP"
{mid1/bot1/inv1 mid1/bot2/inv1}
1

prompt> get_cells -hierarchical -filter "ref_name == IVA"
{mid1/bot1/inv2 mid1/bot2/inv2 mid1/inv3}
1
```

In the previous example, the bot design has cells named inv1 and inv2 that were linked to lsi_10k/IVA. The mid1/bot1 and mid1/bot2 cells instantiate the bot design. The mid1/inv3 cell is also linked to lsi_10k/IVA. After running the **change_link** command in the example, the mid1/bot1/inv1 and mid1/bot2/inv1 cells are relinked to lsi_10k/IVAP, because they are the instances of the same inv1 cell in the bot design. The mid1/bot1/inv2 and mid1/bot2/inv2 cells are not relinked because they are not the instances of the inv1 cell in the bot design. The mid1/inv3 cell is not relinked because it is not in an instance that instantiates the bot design.

The following example uses the change_link command to change the reference of cell using the -pin_map option:

```
prompt> change_link U3 lsi_10k/IVDA -pin_map { {A B} {Z X}}
```

SEE ALSO

- copy_design(2)
- current_design(2)
- link(2)
- rename_design(2)

change_names

Changes the names of ports, cells, and nets in a design.

SYNTAX

```
status change_names
  [-rules name_rules]
  [-hierarchy]
  [-verbose]
  [-names_file names_file]
  [-log_changes log_file]
  [-restore]
  [-dont_touch object_list]
  [-instance instance]
  [-new_name new_name]
  [-skip_inactive_constraints]
```

Data Types

```
name_rules    string
names_file    string
log_file      string
object_list   list
instance      string or instance object
new_name      string
```

ARGUMENTS

-rules *name_rules*

Specifies a name rule set that details the rules to which the object names must conform. The *name_rules* file is defined by using the **define_name_rules** command.

By default, this value is the *name_rules* file specified by the **default_name_rules** variable.

The tool ignores the **-rules** option if you specify the **-names_file** option.

-hierarchy

Modifies all names in the design hierarchy.

By default, the tool changes only objects in the current design.

-verbose

Reports every name change.

By default, the tool provides only a summary of the number of name changes in the design.

-names_file *names_file*

Specifies a file of manual name changes. These name changes are not subject to any name rules, but an error is reported if any name changes are not unique.

By default, the command automatically makes name changes based on the naming rules.

If you specify this option, the tool ignores the **-rules** option.

-log_changes *log_file*

Specifies the log file in which to record all name changes.

If you run multiple **change_names** commands, the *log_file* must be empty before running the first **change_names** command.

-restore

Reverses the changes recorded in the *names_file* during the current session and restores the contents of the file to the state it was in when opened.

-dont_touch *object_list*

Specifies the designs on which the **change_names** command is not to be applied. You can specify any design under the hierarchy of the current design in the object list to ensure that the **change_names** command does not apply any changes to them. The dot (.) character represents the current design.

By default, there are no designs in the list.

-instance *instance*

Specifies the instance on which to apply the **change_names** command.

This option must be used with the **-new_name** option to specify a new instance name.

This option is mutually exclusive with all other options except **-new_name** and **-verbose**.

-new_name *new_name*

Specifies the new instance name when the **-instance** option is used. The new name must consist of only alphanumeric characters and the underscore (_) and must start with an alphabetical character. Specify only the instance name, not a hierarchical name. The new name cannot be a reserved word used by Verilog, SystemVerilog, or VHDL. The new name must not conflict with an existing instance, port, and net name within the same design. The leading backslash will be removed from the new name.

-skip_inactive_constraints

When you specify this option, the inactive constraints (that are part of the non-current design or part of inactive scenarios) are not preserved when you use the **-instance** option with **change_names**. This option has no impact when you do not specify the **-instance** option.

DESCRIPTION

The **change_names** command changes the names of ports, cells (including physical-only cells), and nets in a design to conform to specified name rules.

This command cannot be used to change the name of library cells or bus ports.

To change the name of bus ports, you must first use the **remove_bus** command to remove the bus property from the port.

If an object name does not conform to the specified rules, the tool changes the name and ensures that the new name is unique within the design.

By default, the tool changes the names of bus members to force them to use the same base name as their owning bus. If you do not want the names of bus members, set the **change_names_dont_change_bus_members** variable to **true** before running the **change_names** command.

To show the effects of running this command without actually making the changes, use the **report_names** command.

There are two primary reasons for using the **change_names** command:

- It enables you to modify design object names in the tool so that the names match those that are ultimately created for a saved design. The names the tool displays in reports and in other information match those used in your target system.
- It enables you to define naming rules specific to your target system. For example, you might be using VHDL as a design transfer mechanism, but the naming rules of your system might be more restrictive than those supported by the true VHDL format.

To obtain a list of available name rules, use the **report_name_rules** command. For information about naming rules that can be affected during the execution of the **change_names** file, see the **define_name_rules** man page.

When you run the **change_names** command with no options, it operates on the ports, cells, and nets in the current design. When you specify the **-hierarchy** option, changes are expanded to include all design objects within the current design hierarchy.

For runtime reasons, it's best to use the **-instance** option only when you have a small list of instances to be renamed. If you have a big list of instances to be renamed, you must use the **-rule** option with **change_names** or use the **-skip_inactive_constraints** option and apply the constraints after running **change_names -instance**.

Names File Format

To specify a name change, you must specify the following four fields:

```
design_name object_type object_name new_name
```

The *design_name* field is a design currently read into the tool. The *object_type* field is **port**, **cell**, or **net**. The *object_name* field is the name of an object within the specified design. The *new_name* field is the name that replaces the existing name.

If the file contains a report bar formed by a dashed line (-----), the tool ignores all information above the report bar. The tool parses all lines after the report bar (or all lines, if the file does not contain a report bar).

By default, the tool parses each line into four fields, which are separated by whitespace characters, such as blanks, tabs, or new lines.

This format matches the format of the **report_names** command output, which allows you to redirect the output of the **report_names** command to a file, edit it, and then use it as input to the **change_names** command.

The tool also supports the use of a semicolon (;) as a delimiter to separate the name change specifications in the file. If you use a delimiter, you must add the following line above the report bar formed by the dashed line (-----):

```
use delimiter:true
```

The following example shows a names file without a delimiter:

```
*****
Report : names
       -rules MY_RULES
Design : TOP
Version: v3.0
Date   : Tue Aug 13 14:24:23 2007
```

Design	Type	Object	New Name
TOP	cell	U\$1	U_1
TOP	net	NET_NAME_IS_WAY_TOO_LONG	NET_NAME_IS_WAY_TOO
TOP	net	12345	N12345

The following example shows a names file with a delimiter:

use delimiter:true

Design	Type	Object	New Name
TOP	cell	U\$1	U_1 ;
TOP	net	NET_NAME_IS_WAY_TOO_LONG	NET_NAME_IS_WAY_TOO ;
TOP	net	12345	N12345 ;

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to change the names in the current design to conform to the rules defined by the **default_name_rules** variable:

```
prompt> list default_name_rules
default_name_rules = "EXAMPLE"
```

```
prompt> change_names
Information: Using name rules 'EXAMPLE'.
Information: 4 names changed in design 'TOP'.
```

The following example shows how to use the **-verbose** option to show each name changed by the **change_names** command.

```
prompt> change_names -verbose
Information: Using name rules 'EXAMPLE'.
```

Design	Type	Object	New Name
TOP	port	A1	a_
TOP	port	A2	a_a
TOP	port	A3	a_b
TOP	port	A4	a_c

The following example shows how to use the **-hierarchy** option to change names throughout the design hierarchy. Typically, you want to change the names of all designs in the hierarchy to conform to your name rules.

```
prompt> change_names -hierarchy
Information: Using name rules 'EXAMPLE'.
Information: 12 names changed in design 'HALF_ADDER'.
```

Information: 35 names changed in design 'SUBTRACTOR'.
 Information: No names changed in design 'TOP'.

The following example shows how to use the **define_name_rules** command to create a set of simple name rules. In this example, the name rules require that all names use uppercase characters or numerals.

```
prompt> define_name_rules CAPS_ONLY -allow "A-Z 0-9"
```

```
prompt> change_names -rules CAPS_ONLY -verbose
```

Information: Using name rules 'CAPS_ONLY'.

Design	Type	Object	New Name
MY_BUFFER	port	a1	A1
MY_BUFFER	port	a2	A2
MY_BUFFER	port	b1	B1
MY_BUFFER	port	b2	B2
MY_BUFFER	cell	u1	U1
MY_BUFFER	cell	u2	U2

The following example shows how to make manual name changes to design objects by using the **report_names** and **change_names** commands. First, redirect a report of all the original design objects to a file and edit the file to make manual changes. Use the edited names file to direct the name changes made by the **change_names** command. In this case, the tool ignores the name rules and makes only the specified changes.

```
prompt> report_names -original > TOP.names
/* Edit the TOP.names names file */
prompt> change_names -names_file TOP.names
```

Information: 15 names changed using names file 'TOP.names'

The following example shows how to use the **-instance** option to change one instance name:

```
prompt> change_names -instance a1 -new_name a2
```

The following example shows how to rename a bus port by first removing the bus property and then using the **change_names** command to change the name.

```
prompt> remove_bus -type port -name my_bus
prompt> change_names -names_file name.change -verbose
```

SEE ALSO

define_name_rules(2)
 report_name_rules(2)
 report_names(2)
 default_name_rules(3)

change_selection

Changes the selection in the GUI, taking a collection of objects and changing the selection according to the type of change specified.

SYNTAX

```
status change_selection
  [-name slct_bus]
  [-replace]
  [-add]
  [-remove]
  [-toggle]
  [-type object_type]
  [-clock_trees clock_tree_list]
  collection
```

Data Types

<i>slct_bus</i>	string
<i>object_type</i>	list
<i>clock_tree_list</i>	list
<i>collection</i>	list

ARGUMENTS

-name *slct_bus*

Specifies to change the selection bus by using the value of *slct_bus*. By default, the command changes the selection bus by using the name *global*.

-replace

Replaces the current selection with the objects in the collection. This is the default behavior when no options are specified.

-add

Adds the objects in the collection to the current selection. By default, this option is off.

-remove

Removes the objects that are specified in the collection from the current selection. By default, this option is off.

-toggle

Adds each item that is specified in the collection to the selection bus if it is not currently contained in the selection bus. If it is currently contained in the selection bus, it is removed. By default, this option is off.

-type *object_type*

Specifies the type to change. Only those items from the collection that are of the type specified by *object_type* are used to change the selection. The valid values are **design**, **port**, **net**, **cell**, **pin**, and **path** (timing path). By default, the command uses the entire collection.

-clock_trees *clock_tree_list*

Specifies a list of clock trees for changing the selection. By default, this option is off.

collection

Specifies the collection of objects to use to change the selection. The type of change that is applied to the current selection with the *collection* is specified by the options listed above. By default, this option is off.

DESCRIPTION

The **change_selection** command changes the selection in the GUI. When selections are changed, the GUI updates all relevant windows to reflect it.

A collection of objects and the type of change are given as input to the command. The collection of objects might be returned as the result of another command, such as the **get_designs** command. If the collection is empty and you use the **-replace** option (or let the command default by specifying no option), the current selection is cleared.

If you use the **-type** option, only the type of objects specified are used to change the current selection. For example, if you use **-type design**, the command uses only the design objects in the collection to change the current selection. If you do not use the **-type** option, all objects in the collection are used to change the current selection. For example, if you use the **-add** option without using the **-type** option, all objects, regardless of their type, are added to the current selection.

For information about collections, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example replaces the current selection with the collection of design objects, regardless of type:

```
prompt> change_selection [get_designs]
```

The following example adds a cell named U5 to the selection made in the example above:

```
prompt> change_selection -add [get_cells U5]
```

The following example removes all pin objects from the current selection:

```
prompt> change_selection -remove -type pin [get_selection]
```

The following example clears the selection:

```
prompt> change_selection ""
```


The following example creates a new selection bus and adds a net named n1 to the selection:

```
prompt> set slct [create_selection_bus]
```

```
prompt> change_selection -name $slct [get_nets 1]
```

SEE ALSO

- [collections\(2\)](#)
- [filter\(2\)](#)
- [filter_collection\(2\)](#)
- [get_selection\(2\)](#)
- [query_objects\(2\)](#)

characterize

Captures information about the environment of specific cell instances and assigns the information as attributes on the design to which the cells are linked.

SYNTAX

```
status characterize  
  cell_list  
  [-no_timing]  
  [-constraints]  
  [-connections]  
  [-power]  
  [-verbose]
```

Data Types

cell_list list

ARGUMENTS

cell_list

Specifies the cells in the current design whose designs are to be characterized.

-no_timing

Indicates not to characterize the timing characteristics of the design. Attributes representing the timing environment or requirements are not placed on the subdesigns.

-constraints

Places area, power, connection class, and design rule constraint information on the subdesigns.

-connections

Characterizes the connection attributes of the design.

-power

Uses the switching activity information of the specified cells to annotate the corresponding subdesigns.

-verbose

Echoes the equivalent commands that are applied to the subdesign being characterized.

DESCRIPTION

The **characterize** command places on a design the information and attributes that characterize its environment in the context of a specified instantiation in another design.

The primary purpose of the **characterize** command is to capture the timing environment of the subdesign. This occurs if the **characterize** command is run without the **-no_timing** option.

The command derives and asserts the following on the design to which the instance is linked:

- Unless the **-no_timing** option is used, the **characterize** command places on the subdesigns any timing characteristics previously set by the following commands:

```
create_clock
group_path
read_sdf
read_clusters
set_annotated_check
set_annotated_delay
set_auto_ideal_net
set_disable_timing
set_drive
set_driving_cell
set_false_path
set_ideal_latency
set_ideal_net
set_ideal_transition
set_input_delay
set_load
set_max_delay
set_min_delay
set_multicycle_path
set_operating_conditions
set_output_delay
set_resistance
set_timing_ranges
set_wire_load_model
set_wire_load_mode
set_wire_load_selection_group
set_wire_load_min_block_size
```

- If the **-constraint** option is used, the **characterize** command places on the subdesigns any area, power, connection class, and design rule constraints previously set by the following commands:

```
set_cell_degradation
set_connection_class
set_dont_touch_network
set_fanout_load
set_fix_multiple_port_nets
set_ideal_network
set_ideal_net
set_max_area
set_max_capacitance
set_max_fanout
set_max_power
set_max_transition
```

set_min_capacitance
set_min_porosity

- If the **-connections** option is used, the **characterize** command places on the subdesigns any connection attributes previously set by the following commands:

set_equal
set_logic_one
set_logic_zero
set_logic_dc
set_opposite
set_unconnected

Note that connection class information is applied only if the **-constraint** option is used.

- If the **-power** option is used, the **characterize** command places on the subdesigns any switching activity information, toggle rates, and static probability previously set, calculated, or saved by the following commands:

report_power
set_switching_activity

In most cases, characterizing a design removes the effects of any previous characterization and replaces all relevant information. However, in the case of back-annotation (the **set_load**, **set_resistance**, **read_sdf**, **set_annotated_delay**, an **set_annotated_check** commands), the annotations are moved during the characterize step, and cannot overwrite existing annotations made on the subdesign. In this case, annotations must be explicitly removed from the subdesign by using the **reset_design** command before running the **characterize** command the next time.

The **characterize** command can be used with **set_dont_touch** to maintain the hierarchy during optimization. This method of optimization is known as bottom-up optimization. It can be applied using a golden instance or a uniquify approach. An alternative to bottom-up optimization is top-down optimization, otherwise known as hierarchical compile. In top-down optimization, the tool performs characterization and optimization of subdesigns automatically.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows the golden instance approach to bottom-up optimization. This is useful when the same design is used in several places, with a similar environment. The **characterize** and **set_dont_touch** commands are used to optimize the subdesign once and reference that optimized design in several places. The characterization is done using a golden instance (U1 in this case) and reflects the environment of that instance. Assume that U1 and U2 both reference the design named ADDER and have similar environments.

Characterize the ADDER design with respect to instance "U1" and compile it:

```
prompt> current_design TOP
prompt> characterize U1
prompt> current_design ADDER
prompt> compile
```

Compile the top level while preserving "U1" and "U2":

```
prompt> current_design TOP
prompt> set_dont_touch {U1 U2}
```

```
prompt> compile
```

The following example shows the **uniquify** approach to bottom-up optimization. If a design is used in more than one place with different environments, use the **uniquify** command to create a design for each instance. You can now characterize the new designs and optimize each separately.

Uniquify the hierarchy so that all hierarchical cells reference different designs. This creates new design names such as **ADDER_1** and **ADDER_2**:

```
prompt> current_design TOP
prompt> uniquify
```

Characterize the unique designs with respect to their environments, and compile each design:

```
prompt> characterize {U1 U2}
prompt> current_design ADDER_1
prompt> compile
prompt> current_design ADDER_2
prompt> compile
```

Compile the top-level design while preserving U1 and U2:

```
prompt> current_design TOP
prompt> set_dont_touch {U1 U2}
prompt> compile
```

The following example shows top-down optimization. In top-down optimization, the tool implicitly performs characterization for each subdesign.

```
prompt> current_design TOP
prompt> uniquify
prompt> compile
```

SEE ALSO

```
compile(2)
create_clock(2)
current_design(2)
group_path(2)
link(2)
read_sdf(2)
remove_annotated_check(2)
remove_annotated_delay(2)
remove_constraint(2)
reset_design(2)
set_annotated_check(2)
set_annotated_delay(2)
set_cell_degradation(2)
set_connection_class(2)
set_disable_timing(2)
set_dont_touch(2)
set_dont_touch_network(2)
set_drive(2)
set_driving_cell(2)
set_equal(2)
```

set_false_path(2)
set_fanout_load(2)
set_fix_multiple_port_nets(2)
set_ideal_latency(2)
set_ideal_net(2)
set_ideal_network(2)
set_ideal_transition(2)
set_input_delay(2)
set_load(2)
set_logic_one(2)
set_logic_zero(2)
set_logic_dc(2)
set_min_capacitance(2)
set_max_area(2)
set_max_capacitance(2)
set_max_delay(2)
set_max_fanout(2)
set_max_transition(2)
set_min_delay(2)
set_multicycle_path(2)
set_operating_conditions(2)
set_opposite(2)
set_output_delay(2)
set_resistance(2)
set_target_library_subset(2)
set_timing_ranges(2)
set_unconnected(2)
set_wire_load_model(2)
set_wire_load_mode(2)
set_wire_load_selection_group(2)
set_wire_load_min_block_size(2)
uniquify(2)

check_bindings

Checks the bindings in a synthetic library module definition.

SYNTAX

```
status check_bindings
  [-bindings binding_list]
  [-pin_widths pin_width_list]
  module_name
```

Data Types

```
binding_list    list
pin_width_list  list
module_name     string
```

ARGUMENTS

-bindings *binding_list*

Lists the bindings of the module to check. The default is to check all bindings.

-pin_widths *pin_width_list*

Lists the pin width that is checked for the *bound_operator*. The default is specified by the operator's **check_pin_widths** attribute. The **-pin_widths** option can be used only if all specified bindings bind to the same operator.

module_name

Specifies the name of the synthetic library module whose bindings you want to check. This parameter is required.

DESCRIPTION

The **check_bindings** command ensures that bindings from a synthetic library module to synthetic library operators are correct.

By default, **check_bindings** checks every binding of the specified module. The **check_bindings** command verifies that a binding implements a particular operator correctly. The binding under test is correct if it generates a design that is functionally the same as the "golden" binding. All Synopsys operators are associated with thoroughly-tested golden bindings.

To check a binding for a particular operator, that operator must be annotated with a module and binding known to be correct. Do this using the **check_module** and **check_binding** attributes. For example:

```

operator (ADD_UNNS_OP) {
  check_module : "add";
  check_binding : "b1";
  check_pin_widths : "A=1,B=1,Z=1; A=3,B=4,Z=5";
  /* other declaration go here */
}

```

By default, each binding to an operator is verified for a number of different operator pin widths. The default pin widths are specified by a **check_pin_widths** attribute on the operator. The **check_pin_widths** attribute is a list of pin-width specifications, separated by semicolons (;). Each pin-width specification is a comma (,) separated list of pin sizes. Pin sizes are specified by a pin name, followed by an equal sign (=), followed by a pin width.

The preceding example runs one verification for operators with all pins of width 1 and a second verification with the pin A pin of width 3, pin B of width 4, and pin Z of width 5.

LIMITATIONS

Because **check_bindings** is based on **check_design**, it inherits the same limitations. The **check_bindings** command does not verify noncombinational parts unless the flip-flop names are identical. The **check_bindings** command also does not check very large multipliers because the verification is too expensive.

EXAMPLES

The following example checks the bindings to the SYNTH_ADD module:

```
prompt> check_bindings SYNTH_ADD
```

The next example checks binds b1 and b2 for specific pin widths. Note that **-pin_widths** lists are specified in the list format of dc_shell, not as a single string (as used in the previous **check_pin_widths** attribute).

```

prompt> check_bindings
-binding { b1 b2 }
-pin_widths { "A=2,B=2,Z=2" "A=31,B=31,Z=32" }
SYNTH_ADD

```

SEE ALSO

check_implementations(2)
synthetic_library(3)

check_block_abstraction

Checks the readiness of a block abstraction for usage in a top-level design at various stages of the flow.

SYNTAX

status **check_block_abstraction**

ARGUMENTS

The **check_block_abstraction** command has no arguments.

DESCRIPTION

The **check_block_abstraction** command checks the blocks linked to the top-level design to ensure they are ready for the top-level compile flow. The following general checks are performed:

- Scenario consistency in multicorner-multimode blocks.
- The existence of **dont_touch** attribute settings on blocks and cells.

If any error messages are reported by this command, you should resolve them before proceeding further. If any warning messages are reported by this command, you should review them before proceeding further.

Multicorner-Multimode Support

This command uses information from active scenarios only.

EXAMPLES

The following example checks the blocks in the current design and reports the errors to be resolved:

```
prompt> check_block_abstraction
Error: No scenario data loaded from block reference 'blk1'. (ILM-151)
Error: dont_touch attribute incorrectly set to false on block design 'blk2'. (ILM-121)
0
```

SEE ALSO

`create_block_abstraction(2)`
`set_top_implementation_options(2)`
`report_block_abstraction(2)`

check_bsd

Checks whether a design's boundary-scan implementation is compliant with IEEE Std 1149.1.

SYNTAX

```
status check_bsd  
  [-verbose true | false]  
  [-infer_instructions true | false]  
  [-effort low | medium | high]
```

ARGUMENTS

-verbose true | false

Generates multiple warning or error messages for a set of similar violations during a single step when set to **true**. By default, a warning or error message is generated for only the first violating cell or port.

-infer_instructions true | false

Specifies whether **check_bsd** should analyze the logic to identify the implemented instructions.

When set to **false**, the tool uses only instruction information that was previously specified with the **set_bsd_instruction** command. In the insertion flow, this is provided by **set_bsd_instruction -view spec** commands for specified instructions and by the tool for automatically implemented mandatory instructions. In the existing-logic flow, this is provided with the **set_bsd_instruction -view existing_dft** command. The **check_bsd** command reports this as follows:

...Using the implemented instructions information.

When set to **true**, the tool uses logic analysis to identify the implemented instructions, with an analysis effort optionally specified by the **-effort** option. The **check_bsd** command reports this as follows:

...Inferring the implemented INTEST instruction.
...Inferring the implemented CLAMP instruction.
...Inferring the implemented HIGHZ instruction.
...Inferring the implemented IDCODE & USERCODE instruction.
...Inferring the implemented RUNBIST instruction.

If any instructions have been specified with the **set_bsd_instruction** command, the default is **false**. If no instructions have been specified, the default is **true**.

When this option is set to **true**, note the following:

- Any instruction information specified with the **set_bsd_instruction** command is ignored. If you write out a BSDL file, nonstandard (user-defined) instructions will have USER<n> names and all user-defined test data registers will have UTDR<n> names.

- All unused encodings are assigned to the BYPASS instruction in the BSDL file.

-effort low | medium | high

Controls the effort used to search for implemented instructions when the **-infer_instructions** option is set to **true** and the instruction register width is 16 or greater. The effort levels are:

- **low** - uses only heuristics
- **medium** - uses heuristics first, then random opcode generation for limited sequential search
- **high** - uses a full sequential search

If the **-infer_instructions** options is set to **true** and the instruction register width is less than 16, this option is ignored and a full sequential search (the **high** effort level) is used.

Instruction register masking can be used to reduce search runtime. See the DESCRIPTION section for more information.

DESCRIPTION

The **check_bsd** checks whether the current design's boundary-scan implementation is compliant with IEEE Std 1149.1. You must specify the Test Access Ports (TMS, TCK, TRST, TDI, and TDO) using the **set_dft_signal -view existing** command. If violations to any of the IEEE Std 1149.1 rules are found, the appropriate messages are generated. Check the design for any unresolved references before using this command. Set system clocks using the **set_dft_signal** command in order to see correct functionality. Specify any compliance enable ports using the **set_bsd_compliance** command.

Fatal error messages indicate that your design violates an IEEE Std 1149.1 rule and the violation is serious enough that the compliance checker can no longer continue analysis. You cannot use subsequent tools (for example, the BSDL generator) until you correct the violation.

Error messages indicate that your design violates an IEEE Std 1149.1 rule, but it is still possible to analyze further and gather more information about the design. However, you cannot use subsequent tools (for example, the BSDL generator) until you correct the violation.

Warning messages indicate that your design violates an IEEE Std 1149.1 rule, but the violation does not prevent the BSDL generator from generating BSDL.

When the tool must infer the implemented instructions, you can shorten the sequential search space by masking bits within the IR instruction opcodes using the following two application variables:

- The **test_cc_ir_masked_bits** variable specifies the IR bits to be masked. The compliance checker masks the IR bits that contain "1" in the binary equivalent to the decimal integer value assigned to this variable. For details, see the man page for the **test_cc_ir_masked_bits** variable.
- The **test_cc_ir_value_of_masked_bits** variable specifies a predefined value to be forced onto the masked IR bits. The decimal integer value assigned to this variable is converted to binary for the bit width of the IR, and the masked bits are forced with the corresponding values during the search operation. For details, see the man page for the **test_cc_ir_value_of_masked_bits** variable.

The least significant bit of the IR is considered to be the bit closest to the TDO port.

By default, compliance check is done in accordance with the IEEE1149.1-2001 standard version. You can switch back to the IEEE1149.1-1993 standard version, by setting the **-ieee1149.1_1993** option of the **set_bsd_configuration** command.

For more details about the IEEE Std 1149.1 rules, refer to the *IEEE Std Test Access Port and Boundary-Scan Architecture*.

EXAMPLES

The following command performs compliance checking for the current design, and illustrates the types of messages that are output:

```
prompt> current_design my_boundary_scan

prompt> set_dft_signal -view existing -port my_tdi -type TDI

prompt> set_dft_signal -view existing -port my_tdo -type TDO

prompt> set_dft_signal -view existing -port my_tms -type TMS

prompt> set_dft_signal -view existing -port my_trst -type TRST -active_state 0

prompt> set_dft_signal -view existing -port my_tck -type TCK -timing {45 55}

prompt> set_bsd_compliance -name p1 \
    -pattern {my_compliance_port, 0}

prompt> set_dft_signal -view spec -port my_system_clock \
    -type ScanMasterClock -timing {1 2}

prompt> check_bsd -v true
Loading target library 'lsi_10k'
Loading design 'my_boundary_scan'
...Starting IEEE 1149.1 Compliance Checking.
...Finding set of sequential elements.
...Analyzing TAP and TAP Controller.
.....Analyzing TAP.
.....Finding the set of TAP controller sequential elements.
.....Pruning set of TAP Controller sequential Elements
...Checking the TAP controller initialization.
...Analyzing the TAP controller reset condition.
...Traversing the TAP controller states.
...Inferring TAP controller clock outputs.
...Analyzing TRST port.
Warning: Undriven input port TRST is floating. When undriven,
this port should behave as though it was driven by logic one. (TEST-819)
...Analyzing TMS Port.
Warning: Undriven input port TMS is floating. When undriven,
this port should behave as though it was driven by logic one. (TEST-819)
...Analyzing TCK halt state.
Warning: Undriven input port TDI is floating. When undriven,
this port should behave as though it was driven by logic one. (TEST-819)
...Analyzing the instruction register.
.....Finding the set of Shift Elements forming the shift register
.....Finding the update flops.
...Analyzing the BYPASS register.
.....Finding the set of Shift Elements forming the shift register
.....Finding the set of Synchronous Shift Elements forming the shift register
Error: Illegal capture value of BYPASS register. (TEST-881)
...Analyzing the DIR register.
.....Finding the set of Shift Elements forming the shift register
...Analyzing the boundary-scan register.
```

.....Finding the set of Shift Elements forming the shift register
Finding the update flops.
Finding the BSR cells controlling the design ports.
Finding the BSR cells sensing the design ports.
Finding the BSR cells driving the design ports.
 ...Finding the set of IR decoding pins.
 ...Analyzing the different signatures at the decoding pins.
 ...Inferring the different test data registers selected by instructions.
 ...Checking output conditioning of the implemented instructions.
 ...Inferring the implemented SAMPLE/PRELOAD instructions.
 ...Finding the BSR controlling cells PIs.
 ...Finding the BSR controlling cells POs.
 ...Finding the BSR non-controlling cells PIs.
 ...Finding the BSR cells POs.
 ...Inferring the implemented INTEST instruction.
 ...Inferring the implemented CLAMP instruction.
 ...Inferring the implemented HIGHZ instruction.
 ...Inferring the implemented IDCODE & USERCODE instruction.
 ...Inferring the implemented RUNBIST instruction.
 ...Analyzing the EXTEST instruction.
 ...Analyzing the BYPASS instruction.
 ...Analyzing the SAMPLE/PRELOAD instruction.
 ...Analyzing the INTEST instruction.
 ...Analyzing the RUNBIST instruction.
 ...Analyzing the IDCODE and USERCODE instructions.

IEEE 1149.1 Summary

4 state elements found in the TAP controller
 5 cells found in the Instruction Register
 5 standard instructions found.
 11 user defined instructions found.
 1 cells in BYPASS register
 32 cells in DEVICE ID register
 5 cells in boundary-scan register

IEEE 1149.1 Violation Summary

6 Violations found in extraction of TAP Controller
 Violates Rules: 3.3.1b 3.6.1b
 Violates Rules: 3.3.1b 3.6.1b
 Violates Rules: 3.3.1b 3.6.1b
 1 Violations found in extraction of BYPASS register
 Violates Rules: 9.1.1b

SEE ALSO

```
set_bsd_compliance(2)  
set_dft_signal(2)
```

check_budget

Checks that user-specified budgets and fixed delays are consistent with path constraints.

SYNTAX

```
status check_budget
  [-verbose]
  [-tolerance tolerance]
  [-from object_list]
  [-to object_list]
  [-no_interblock_logic]
  [cell_list]
```

Data Types

```
tolerance    float
object_list  list
cell_list    list
```

ARGUMENTS

-verbose

Shows the path segments that cause violation of timing constraints. By default, only the number of paths that violate timing constraints are shown.

-tolerance *tolerance*

Specifies that only paths with a slack greater than *tolerance* in absolute value are checked and reported. By default, all paths are checked and reported.

-from *object_list*

Shows only the paths from the named pins, ports, or endpoints clocked by named clocks. By default, all paths that violate timing constraints are shown.

-to *object_list*

Shows only the paths to the named pins, ports, or endpoints clocked by named clocks. By default, all paths that violate timing constraints are shown.

-no_interblock_logic

Specifies that interblock logic is not to be budgeted. By default, the tool considers interblock logic not fixed.

cell_list

Specifies a list of cells to budget. This option must be set only when the **check_budget** command is used before the **allocate_budget** command.

DESCRIPTION

The **check_budget** command checks that the user-specified budgets and fixed delays are consistent with path constraints. Inconsistencies exist when the sum of user-specified budgets and fixed delays along a path exceed the value of the path constraint. By default, the report shows only the number of paths that violate the timing constraints. The **-verbose** option shows details about the violations. The *cell_list* option allows you to check user budgets on the specified cells before budget allocation. This option must not be used after budget allocation.

EXAMPLES

The following example shows a report using only the default values:

```
prompt> check_budget
```

```
*****
Check budget
*****
Design has 2 violations.
1
```

The following example shows a report with the verbose option.

```
prompt> check_budget -verbose
```

```
*****
Check budget
*****

Startpoint: C3/OUT_reg (rising edge-triggered flip-flop clocked by clk)
Endpoint: C3/OUT_reg (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: max
```

Point	Incr	Path
clock clk (rise edge)	0.00	0.00
clock network delay (ideal)	0.00	0.00
C3/OUT_reg/CLK (fdmf1i2)	0.00	0.00 r
C3/OUT_reg/Q (fdmf1i2)	0.00 *	0.00 r
C3/OUT_reg/D1 (fdmf1i2)	0.00	0.00 f
data arrival time	0.00	
clock clk (rise edge)	500.00	500.00
clock network delay (ideal)	0.00	500.00
clock uncertainty	-20.00	480.00
C3/OUT_reg/CLK (fdmf1i2)	0.00	480.00 r
library setup time	-504.81	-24.81

data required time	-24.81

data required time	-24.81
data arrival time	0.00

slack (VIOLATED)	-24.81

Startpoint: IN (input port clocked by clk)
 Endpoint: C1/rOUT_reg
 (rising edge-triggered flip-flop clocked by clk)
 Path Group: clk
 Path Type: max

Point	Incr	Path

clock clk (rise edge)	0.00	0.00
clock network delay (ideal)	0.00	0.00
input external delay	250.00	250.00 f
C1/IN (D1)	0.00	250.00 f
C1/rOUT_reg/D0 (fdmf1c2)	0.00	250.00 f
data arrival time	250.00	
clock clk (rise edge)	500.00	500.00
clock network delay (ideal)	0.00	500.00
clock uncertainty	-20.00	480.00
C1/rOUT_reg/CLK (fdmf1c2)	0.00	480.00 r
library setup time	-342.00	138.00
data required time	138.00	

data required time	138.00	
data arrival time	-250.00	

slack (VIOLATED)	-112.00	

Design has 2 violations.
 1

SEE ALSO

report_path_budget(2)
 set_user_budget(2)

check_design

Checks the current design for consistency.

SYNTAX

```
status check_design
  [-summary]
  [-no_warnings]
  [-one_level]
  [-multiple_designs]
  [-no_connection_class]
  [-nosplit]
  [-unmapped]
  [-cells]
  [-ports]
  [-designs]
  [-nets]
  [-tristates]
  [error-ids]
  [-html_file_name html_file]
```

ARGUMENTS

-summary

Displays a summary of the warning messages instead of one message per warning. This does not affect the way error messages are issued.

-no_warnings

Suppresses warning messages, so only error messages are printed.

-one_level

Performs checks at only the current level of hierarchy. By default, **check_design** checks the current level and all designs below the current level.

-multiple_designs

Reports warning messages related to multiply-instantiated designs. With this option, the tool lists all multiply-instantiated designs along with instance names and associated attributes, such as **dont_touch**, **black_box**, and **ungroup**. By default, these messages are suppressed.

-no_connection_class

Suppresses connection class warning messages. This switch is useful while working on GTECH designs and netlists on which

connection class violations are expected. By default, these messages are reported.

-nosplit

Prevents lines from being split when column fields overflow. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line in the correct column.

-unmapped

Reports warning messages for unmapped cells when the cells are being checked. By default, these messages are suppressed.

-cells

Reports the following warning messages for cells: LINT-0, LINT-1, LINT-10, LINT-32, LINT-33, LINT-58, LINT-59, LINT-60, and LINT-61 (if the **-unmapped** option is used).

-ports

Reports the following warning messages for ports: LINT-5, LINT-6, LINT-8, LINT-28, LINT-29, LINT-31, and LINT-52.

-designs

Reports the following warning messages for designs: LINT-25, LINT-46, and LINT-55.

-nets

Reports the following warning messages for nets: LINT-2, LINT-3, LINT-4, LINT-35, LINT-38, LINT-47, and LINT-54.

-tristates

Reports the following warning messages for tristates: LINT-34 and LINT-63.

error-ids

Reports warning messages for specified error IDs, such as LINT-2.

-html_file_name *html_file*

Generates a report for the **check_design** command in HTML format. By default, the **check_design** command writes reports in standard output. The **-html_file_name** option redirects the output to a specified HTML file.

DESCRIPTION

The **check_design** command checks the internal representation of the current design for consistency, and issues error and warning messages as appropriate.

Error messages indicate design problems of such severity that the **compile** command does not accept the design; for example, recursive hierarchy in a design (when a design references itself) is an error. Warning messages are informational and do not necessarily indicate design problems. However, these messages should be investigated.

The **check_design** command flags multiple instances of a design within a system. If a design is instantiated in two different designs, a warning message is issued. For information on how to respond to error messages dealing with multiple instances, see the **uniquify** command man page.

The **check_design -summary** command automatically runs on every design that is compiled. However, you can use the **check_design** command explicitly to see warning messages.

Potential problems detected by this command include unloaded input ports or undriven output ports, nets without loads or drivers or

with multiple drivers, cells or designs without inputs or outputs, mismatched pin counts between an instance and its reference, tristate buses with non-tristate drivers, wire loops (timing loops with no cells in them) across hierarchies, and so forth.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command checks the current design for problems and issues error and warning messages:

```
prompt> check_design
```

The following command checks only the current level of the design:

```
prompt> check_design -one_level
```

SEE ALSO

- check_library(2)
- check_timing(2)
- compile(2)
- current_design(2)
- set_boundary_optimization(2)
- set_dont_touch(2)
- uniquify(2)
- check_design_allow_non_tri_drivers_on_tri_bus(3)
- check_design_check_for_wire_loop(3)

check_error

Reports extended information on message IDs from the last command.

SYNTAX

```
status check_error
  [-verbose]
  [-reset]
```

ARGUMENTS

-verbose

Displays the list of message IDs found during previous commands.

By default, the command only returns the error status.

-reset

Resets the list of previously found message IDs. After resetting the error list, the **check_error -verbose** command returns an empty list.

By default, the command does not modify the message list.

DESCRIPTION

The **check_error** command is used to compare error, warning, or informational messages issued by previous commands to the list of message IDs specified by the **check_error_list** variable. While it is most commonly used for error messages, the variable may contain error, warning, or informational message IDs.

If the **check_error** command finds message IDs that are specified in the **check_error_list** variable, the command returns 1. If the **check_error** command does not find any messages that are specified by the **check_error_list** variable, the command returns 0.

Messages that are suppressed by the **suppress_message** command are reported by **check_error**. If you want suppressed messages to be ignored by **check_error** then you must remove their message IDs from the **check_error_list** variable.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to use the **check_error** command to check if execution errors are specified by the **check_error_list** variable:

```
prompt> printvar check_error_list  
check_error_list = "UID-3 UID-4 EQN-18 EQN-19"
```

```
prompt> check_error  
0
```

```
prompt> link  
Error: Current design is not defined. (UID-4)  
0
```

```
prompt> check_error  
1
```

```
prompt> check_error -verbose  
{UID-4}  
1
```

The following command resets the error list:

```
prompt> check_error -reset -verbose  
{UID-4}  
1
```

```
prompt> check_error -verbose  
0
```

To use the **check_error** command in a script to see if specified errors occurred in the previous commands, use the following command:

```
prompt> if { [check_error] } {  
    echo failure_message  
    exit  
} else {  
    echo success_message  
}
```

SEE ALSO

[check_error_list\(3\)](#)
[suppress_message\(2\)](#)

check_implementations

Checks the implementations in a synthetic library module definition.

SYNTAX

```
status check_implementations
  [-implementations implementation_list]
  [-parameters parameter_list]
  module_name
```

Data Types

```
implementation_list  list
parameter_list      list
module_name         string
```

ARGUMENTS

-implementations *implementation_list*

Lists the implementations of the module to check. The default is all implementations.

-parameters *parameter_list*

Lists the module parameters to use when checking. The default is to use the parameters specified by the **check_params** attribute of the module.

module_name

Specifies the name of the synthetic library module whose implementations you want to check. This argument is required.

DESCRIPTION

The **check_implementations** command ensures that when you add a new implementation to an existing module, the new implementation works in the same manner as the old implementation.

Check implementations are provided by Synopsys for all Synopsys modules. They are never used to implement hardware, only for verification.

```
module (mod1) {
  /* Other declarations go here */
  check_implementation(check_add) {
```



```

    }
    /* other implementations go here */
}

```

By default, every implementation of a module is compared to the verify implementation. For parameterized modules, a verification is performed for each of the **check_params** that is specified for the module.

The **check_params** attribute is a list of parameter specifications, separated by semicolons (;), which specifies a test to be run. Each parameter specification is a comma (,) separated list of parameter names, followed by an equal sign (=), followed by a value. For example, the specification below runs one verification with n=2 and m=3, and a second verification with n=5 and m=8 for each implementation:

```

<examplelast>
check_params : "n=2,m=3; n=5,m=8" ;
<body>

```

check_implementations -impl {rpl cla}add

LIMITATIONS

Because **check_implementations** is based on **check_design**, it inherits the same limitations. The **check_implementations** command does not verify noncombinational parts unless the flip-flop names are identical. The **check_implementations** command also does not check very large multipliers because the verification is too expensive.

EXAMPLES

The following example checks all implementations of the SYNTH_ADD module:

```
prompt> check_implementations SYNTH_ADD
```

This example checks the ripple and carry-lookahead implementations for parameter values of 33 and 63:

```
prompt> check_implementations -impl { rpl_add cla_add } \
-param { "n=33" "n=63" } SYNTH_ADD
```

SEE ALSO

check_bindings(2)
synthetic_library(3)

check_library

Performs consistency checks between logic and physical libraries, across logic libraries, and within physical libraries.

SYNTAX

```
status check_library
  [-logic_library_name logic_library_name_list]
  [-mw_library_name phys_library_name_list]
  [-physical_library_name phys_library_name_list]
  [-cells cell_list]
```

Data Types

```
logic_library_name_list  list
phys_library_name_list  list
cell_list                list
```

ARGUMENTS

-logic_library_name *logic_library_name_list*

Specifies the logic libraries (.db) to be checked. Other types of libraries such as .ILM are not supported.

For logic versus logic library checking, specify two or more libraries, such as the minimum and maximum libraries. If you have set the **-compare** or **-validate** options of the **set_check_library_options** command, specify only two libraries.

If you do not specify this option, the command determines the logic libraries in the following order:

- The minimum and maximum libraries that were set with the **set_min_library** command.
- If you have set the **-scaling** option of the **set_check_library_options** command, the scaling libraries that were set with the **define_scaling_lib_group** command.
- The link libraries that were set with the **link_library** and **search_path** variables.
For logic versus logic library checking, the command first groups the link libraries by cell set, and within each group or family the command checks consistencies across all libraries in the same group. The grouping is by majority (> 50%) of same name cells. For example, if lib1 has 9 cells and lib2 has 6, and 5 of them are the same, then they are grouped together.

This option is ignored for physical library checking.

-mw_library_name *phys_library_name_list*

Specifies Milkyway reference libraries to be checked.

If you do not specify this option, the command uses the reference libraries from the open Milkyway design library. If there is no

open Milkyway design library, the command uses the reference libraries specified in the **mw_reference_library** variable.

-physical_library_name *phys_library_name_list*

Specifies NDM based physical libraries (.frame) to be checked. You can specify multiple physical libraries. For cross checking, specify both logic and physical libraries. If libraries are not specified explicitly in **check_library**, the ones loaded/opened at runtime will be used. The libraries to be checked are in the following priority:

- 1) explicitly specified in **check_library** command
- 2) logic libraries associated with the specified physical library
- 3) loaded or opened logic and physical libraries at runtime -**mw_library_name** and -**physical_library_name** cannot be specified together.

-cells *cell_list*

Specifies a list of cell names to be checked. If not specified, all cells in the libraries are checked.

DESCRIPTION

The **check_library** command checks and reports the items described below, based on the settings of the **set_check_library_options** command.

If you do not run the **set_check_library_options** command and do not specify any options with the **check_library** command, the **check_library** command checks logic versus physical library consistency for the libraries that are set up for the current design. If there are no libraries set up for the current design, the command does not perform any checks.

During logic library checking, to report the delay, power, and noise characterization values in the libraries, the Liberty-format description of the libraries must have the **library_features** attribute set to the respective values: **report_delay_calculation**, **report_power_calculation**, and **report_noise_calculation**.

Use the **check_library** command to verify the libraries before reading in a design.

This command checks library qualities in three main areas:

- Physical library quality
- Logic versus physical library consistency
- Logic versus logic library consistency

All of the checking functions are available in Design Compiler and the functions of logic versus logic library checking are also available in Library Compiler.

Physical Library Quality Checking

Physical library qualities are checked as follows:

- Technology data consistency between the specified main library and each linked reference library
- CEL view versus FRAM view in the library for missing views and mismatched views
- Cells with identical names in different reference libraries and the specified main library
- Signal electromigration rules
- Antenna rules and missing antenna properties

- Rectilinear cells
- Physical-only cells
- Physical properties for place and route including unit tiles for the libraries
- Pin routability
- Technology data quality
- DRC for each cell in the library

Logic Versus Physical Library Consistency Checking

The command performs the following consistency checks between logic and physical libraries:

- Missing cells in the logic or physical library
If you specify multiple logic libraries and no cells are found in any of the libraries that match the physical library, the cells are counted as missing in the logic library unless they are physical-only cells. Otherwise, if the cells are found in any of the logic libraries, such as the first library, they are counted as consistent in cell names.
- Missing or mismatched pins (pin names, directions, and types) in the logic and physical library.
Note that the **input** pin direction keyword in a logic cell versus the **Input** keyword in a physical cell is not a mismatch.
- Area values of each standard cell between the logic (.lib or .db) model and FRAM view (PRBoundary and CellBoundary)
- Cell footprint (**cell_footprint** attribute)
- Bus character naming style in logic and physical libraries.

By default, missing cells and pins and mismatched pins are checked.

For logic versus physical library checking, at the end of checking the command reports a cross-check summary that includes the following information:

- Number of cells missing in the logic library (excluding physical-only cells)
- Number of cells missing in the physical library
- Number of cells with missing or mismatched pins in libraries, if any
- The logic versus physical library quality checking PASSED or the logic library is INCONSISTENT with the physical library

Logic Versus Logic Library Consistency Checking

Logic versus logic library consistency checks include:

- General checking at library, cell, pin, and timing group levels for missing cells, missing and mismatched pins or pg_pins, and timing arcs.
- Special checking for timing, noise, and power scaling
- Special checking for the UPF or multivoltage flow, such as pg_pins, power management cells, and power data
- Special checking for the multicorner-multimode flow, such as operating conditions and power-down functions
- Library characterization and validation

For logic versus logic library checking, at the end of checking the command reports a summary for each specified option. For example, for the **-mcmm** option, it reports:

Logic library consistency check FAILED for MCMM.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, the lib1.db and lib2.db logic libraries are checked against the phys_lib physical libraries for missing cells and pins, mismatched pins, the cell footprint, and the bus delimiter:

```
prompt> set_check_library_options -cell_footprint -bus_delimiter
1
```

```
prompt> check_library -mw_library_name {phys_lib} \
  -logic_library_name {lib1.db lib2.db}
```

```
#BEGIN_XCHECK_LIBRARY
```

```
Logic Library: lib1
               lib2
```

```
Physical Library: phys_lib
```

```
check_library options: -cell_footprint -bus_delimiter
```

```
Version: E-2010.12
```

```
Check date and time: Tue Sep 28 20:34:26 2010
```

List of logic library and file names

```
-----
Logic library name      Logic library file name
-----
```

```
lib1                    /usr/lib/lib1.db
```

```
lib2                    /usr/lib/lib2.db
-----
```

```
#BEGIN_XCHECK_LOGICCELLS
```

```
Number of cells missing in logic library: 3 (out of 3348)
```

```
Information: List of cells missing in logic library (LIBCHK-210)
```

```
-----
Cell name              Cell type      Physical library
-----
```

```
AND2                   Core          phys_lib
```

```
NOR1                   Core          phys_lib
```

```
XOR3                   Core          phys_lib
-----
```

List of physical only cells

```
-----
Cell name              Cell type      Physical library
-----
```

```
GFILL                  Filler        phys_lib
```

```
GFILL10                Filler        phys_lib
```

```
FILL8                  Filler        phys_lib
-----
```

#END_XCHECK_LOGICCELLS

#BEGIN_XCHECK_PHYSICALCELLS

Number of cells missing in physical library: 0 (out of 846)

#END_XCHECK_PHYSICALCELLS

#BEGIN_XCHECK_PINS

Number of cells with missing or mismatched pins in libraries: 1

Error: List of pins mismatched in logic and physical libraries (LIBCHK-213)

Logic library: lib1

Physical library: phys_lib

Cell name	Pin direction		Pin type		
	Pin name	Logic	Physical	Logic	Physical
pn1123	VSSO	output	Output	signal	ground
	SGND	input	Input	signal	ground

#END_XCHECK_PINS

#BEGIN_XCHECK_BUS

Information: List of bus naming styles (LIBCHK-214)

Library name	Library type	Bus naming style
phys_lib	Physical library	_%d>
lib1	Logic library	

#END_XCHECK_BUS

#BEGIN_XCHECK_FOOTPRINT

Number of footprints: 1

Warning: List of cells with cell_footprint attribute (LIBCHK-215)

Footprint	Logic library name	Cell name	PR boundary
TIEH	lib1	GTIEH	(0,0)(0.8,1.8)
	lib1	TIEH	(0,0)(0.6,1.8)

#END_XCHECK_FOOTPRINT

Logic vs. physical library check summary:

Number of cells missing in logic library: 3

Number of cells with missing or mismatched pins in libraries: 1

Information: Logic library is INCONSISTENT with physical library. (LIBCHK-220)

#END_XCHECK_LIBRARY

0

SEE ALSO

open_mw_lib(2)
report_check_library_options(2)
set_check_library_options(2)
link_library(3)
mw_reference_library(3)

check_license

Checks the availability of a license for a feature.

SYNTAX

```
status check_license  
      feature_list
```

Data Types

```
feature_list  list
```

ARGUMENTS

feature_list

Specifies the list of features to be checked. If more than one feature is specified, they must be enclosed in curly braces ({}). By looking at your key file, you can determine all of the features licensed at your site.

DESCRIPTION

The **check_license** command checks on a license for the named features. It does not check out the license.

The **list_licenses** command provides a list of the features that you are currently using.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example checks on the license for the multivoltage feature:

```
prompt> check_license {Galaxy-MV}
```


SEE ALSO

get_license(2)
license_users(2)
list_licenses(2)
remove_license(2)

check_mv_design

Checks for violations in a multivoltage design.

SYNTAX

```
status check_mv_design
  [-verbose]
  [-isolation]
  [-target_library_subset]
  [-opcond_mismatches]
  [-connection_rules]
  [-level_shifters]
  [-power_nets]
  [-clock_gating_style]
  [-objects object_name_list]
  [-models]
  [-max_messages message_count]
  [-output output_file_name]
```

Data Types

<i>message_count</i>	unsigned integer
<i>output_file_name</i>	string

ARGUMENTS

-verbose

Reports the violations in detail. When this option is not used, only a summary of the violations is reported in the log file.

-isolation

Reports electrical isolation issues:

- Nets that require isolation, but the isolation cell is missing
- Nets that do not require isolation, but an isolation cell is present
- Isolation cells that do not cross a power domain boundary

-target_library_subset

Checks for inconsistent settings between the target library, target library subset, and operating conditions. This option reports the following conditions:

- Conflicts between the target library subset and the global **target_library** variable setting. The target library subset must be a

subset of the library set specified by the **target_library** variable.

- Conflicts between the operating condition and target library subset. There must be at the least one library from the target library subset that has the same process, voltage, and temperature as the operating condition being used on a block.
- Conflicts between a cell and the target library subset specification on the parent design of the cell. This check ensures that the cells from the specified target library subset are used.

-opcond_mismatches

Reports the technology cells instantiated in the design with incompatible operating conditions. The option checks for conflicts between a cell and the operating condition specified on the parent design of the cell. This check ensures that the operating condition at which the cell is characterized matches the operating condition specified on the design.

-connection_rules

Reports violations in the always-on synthesis and pass-gate connections:

- Always-on net driven by a normal cell
- Always-on net or a net from an always-on domain driving a pass gate
- Always-on cell driving a normal net
- Two pass gates connected to each other

-level_shifters

Reports level shifter violations:

- Net requiring level-shifting with no level-shifter cell
- Net on which a level-shifter cell cannot be added because either the net is marked as **dont_touch** or the net is driven by a pin operating at a different voltage
- Level-shifter cell shifting an incorrect voltage difference
- Input pin of a level-shifter cell driven by a pin operating at a different voltage
- Output pin of a level-shifter cell driving pins operating at a different voltage
- Level-shifter cell that violates level-shifter strategy settings

-power_nets

Reports power and ground pin connections, including the following information:

- PG pin mismatch between the logical library and FRAM library
- Power and ground connection summary
- Power and ground pins whose connections cannot be derived and the reason that the derivation fails
- Power and ground pins whose existing connections do not match with the derived connection from the power domains.

A mismatch between an existing PG connection and the connection derived from the power domains can occur for any of the following reasons:

- "Unknown power pin type" indicates either the power pin does not have a type or the pin has a type without connection semantics. Automatic power connection supports the following power pin types: `primary_power_pin`, `primary_ground_pin`, `backup_power_pin`, `backup_ground_pin`, `internal_power_pin`, and `internal_ground_pin`.
- "Invalid pin type for the cell or the cell's power domain" indicates that the cell's power domain does not have a power net

connection with the matching type for the power pin type. For example, this situation occurs when a regular cell has a backup ground pin, but the cell's power domain does not have a backup ground net connection.

- "No signal pin that uses this PG pin as the related_power_pin/related_ground_pin." This issue occurs only on level-shifter and isolation cells. The power connection of a power pin on a level-shifter or isolation cell is obtained through tracing cells connected to related signal pins of the power pin. The power connection fails if a power pin has no related signal pin, for example, when no signal pin uses the power pin as the related_power_pin or related_ground_pin in the logical library cell definition.
- "Cells connected to the related signal pins of the level shifter or isolation cell requiring different PG nets." This issue occurs only on level-shifter and isolation cells. The tool cannot derive a proper power net for a specific pin because different nets are needed for the same pin, based on cells connected to the related signal pins.
- "Back-to-back connection of level-shifter and isolation cell" occurs when a level-shifter or isolation cell is directly connected to another level-shifter or isolation cell. The automatic connection of a power pin on such level-shifter or isolation cells might fail if the tracing of related signal connections reaches only other level-shifter or isolation cells.

This option also reports bias pin connections, when a supply net and its connected supply nets through supply ports or power switches are used as the power/nwell/deepnwell net, and also as the ground/pwell/deeppwell net.

-clock_gating_style

Reports hierarchical blocks for which there is no library cell that meets its operating condition and specified clock-gating style. Clock-gate insertion cannot be performed.

-objects *object_name_list*

This option takes a list of pins, or ports as input. It checks level shifting and isolation violations on the paths through the given pins. The option must be specified with one of -level_shifter and -isolation, or both, and it is an error to specify -objects with the options other than -level_shifter, -isolation, or -verbose. The same violation will not be reported more than once.

-models

Reports any modeling errors found in the available library models.

-max_messages *message_count*

Sets a maximum limit on the number of messages written into the log file and output file.

-output *output_file_name*

Writes the output of the **check_mv_design** command to a file specified by the file name argument. If a file of the specified name already exists, it is over-written.

This file can be opened in the **MV Advisor** browser in the **Design Vision** GUI. Alternatively, the same file can be opened in an HTML browser by using the absolute path of the file.

The output file contains the details of all violations reported, even when the **-verbose** option is not used.

DESCRIPTION

The **check_mv_design** command checks the design, multivoltage constraints, electrical isolation requirements, and connection rules. It issues error and warning messages as appropriate.

The checker options can be combined to adjust the level of detail in the final report. If the command is used without any checker options, it reports only a summary of all violations found.

If the **-verbose** option is used, the command reports details of all violations in the log file. The **-max_messages** option limits the number of violations reported. When other checker options are specified, the message count specified by the **-max_messages** option applies to each type of check performed. If no specific checker is specified, the message count applies to the total number of messages generated by all types of checks performed. If **-max_messages** is not specified, the command writes out all messages without limit.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command reports all multivoltage violations in the design:

```
prompt> check_mv_design -verbose
```

```
-----
Target Library Subset Checks
-----
```

No Errors/Warnings Found.

```
-----
Power Domain Checks
-----
```

Warning: Isolation cell is required on net A_INST/OUTPUT connecting A_INST and B_INST. (MV-042)

Warning: Isolation cell is required on net B_INST/OUTPUT connecting B_INST and top. (MV-042)

Warning: Isolation cell A_INST/iso1 is used to isolate a net that is not crossing power domain boundary. (MV-044)

```
-----
Power Domain Checks Summary
-----
```

Warning: Found 2 net(s) without isolation. (MV-046)

Warning: Found 1 isolation cell(s) on net(s) not crossing power domain boundaries. (MV-048)

```
-----
Cell Operating Condition Checks
-----
```

No Errors/Warnings Found.

```
-----
Power Domain and Operating Condition Consistency Checks
-----
```

No Errors/Warnings Found.

The following command generates a summary of all violations in the design:

prompt> **check_mv_design**

Target Library Subset Checks

No Errors/Warnings Found.

Power Domain Checks

Warning: Found 2 net(s) without isolation. (MV-046)

Warning: Found 1 isolation cell(s) on net(s) not crossing power domain boundaries. (MV-048)

Cell Operating Condition Checks

No Errors/Warnings Found.

Power Domain and Operating Condition Consistency Checks

No Errors/Warnings Found.

The following example reports violations related to electrical isolation:

prompt> **check_mv_design -verbose -isolation**

Power Domain Checks

Warning: Isolation cell is required on net A_INST/OUTPUT connecting A_INST and B_INST. (MV-042)

Warning: Isolation cell is required on net B_INST/OUTPUT connecting B_INST and top. (MV-042)

Warning: Isolation cell A_INST/iso1 is used to isolate a net that is not crossing power domain boundary. (MV-044)

Power Domain Checks Summary

Warning: Found 2 net(s) without isolation. (MV-046)

Warning: Found 1 isolation cell(s) on net(s) not crossing power domain boundaries. (MV-048)

The following example reports violations related to power nets:

```
prompt> check_mv_design -verbose -power_nets
```

```
-----  
Power/Ground Pin Connection Checks  
-----
```

```
Error: Supply net sn1 and its connected supply net(s) are used as both  
power/hwell/deepnwell net and ground/pwell/deepnwell net. (MV-601)
```

```
Error: Supply net sn2 and its connected supply net(s) are used as both  
power/hwell/deepnwell net and ground/pwell/deepnwell net. (MV-601)
```

```
Error: Power net hookup for power domains is not specified properly (MV-503)  
-----
```

The following example shows how to use the *-output* option of the **check_mv_design** command.

```
prompt> check_mv_design -verbose -output my_file.cmvd
```

SEE ALSO

- check_library(2)
- compile_ultra(2)
- remove_target_library_subset(2)
- set_operating_conditions(2)
- set_target_library_subset(2)

check_rp_groups

Checks the relative placement constraints and reports any failures.

SYNTAX

```
collection check_rp_groups
  {rp_groups | -all}
  [-output filename]
  [-verbose]
  [-critical]
```

Data Types

rp_groups list or collection
filename string

ARGUMENTS

rp_groups

Specifies the relative placement groups that will be checked for failures. This option and the **-all** option are mutually exclusive.

-all

Specifies that all relative placement groups will be checked. This option and the *rp_groups* argument are mutually exclusive.

-output *filename*

Specifies the name of the output file for the report.

If you do not specify this option, the report is written to the standard output.

-verbose

Reports relative placement cell orientation failures that are not reported otherwise. It also returns more detailed alignment and utilization failure messages that include the exact location details of the failures.

DESCRIPTION

The **check_rp_groups** command checks any violations of relative placement constraints for the specified relative placement groups. The violations include failing to place the specified relative placement groups and the placed relative placement groups not meeting relative placement constraints.

The command returns a collection containing the relative placement groups that are checked. If no groups are checked, an empty string is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples use **check_rp_groups** to check for relative placement failures:

```
prompt> get_rp_groups
{compare::g1 compare::g2 compare::g4 compare::g7}
```

```
prompt> check_rp_groups -all -out failures
{compare::g7 compare::g4 compare::g2 compare::g1}
```

```
prompt> check_rp_groups compare::g4 -out g4_failures
{compare::g4}
```

```
prompt> check_rp_groups compare::g4
```

```
*****
```

Report: The relative placement groups not meeting all constraints but placed.

Version: B-2008.09

Date: Mon Sep 11 04:44:34 2008

Number of relative placement groups: 1

```
*****
```

```
relative placement GROUP: compare::g4
```

```
-----
```

Warning: Possible placement failure of relative placement group 'compare::g4'
(RPGP-027)

```
{compare::g4}
```

```
prompt> check_rp_groups compare::g1
```

```
*****
```

Report: The relative placement groups that could not be placed.

Version: B-2008.09

Date: Mon Sep 11 04:47:25 2008

Number of relative placement groups: 1

```
*****
```

```
relative placement GROUP: compare::g1
```

```
-----
```

Warning: Possible placement failure of relative placement group 'compare::g1'
(RPGP-027)

```
{compare::g14}
```

SEE ALSO

`add_to_rp_group(2)`
`create_rp_group(2)`
`remove_rp_groups(2)`

check_scan_def

Performs scan chain structural consistency checking based either on the scan chain information stored as an attribute in the current .ddc file, or an ASCII SCANDEF file.

SYNTAX

```
status check_scan_def
      [-file file_name]
```

Data Types

file_name string

ARGUMENTS

-file *file_name*

Specifies an optional SCANDEF file containing scan chain information to be used for checking.

If this option is not specified, the command looks for the SCANDEF information stored in the current design database.

DESCRIPTION

The **check_scan_def** command performs a scan chain structural checking based on the scan chain information stored in the current .ddc input file. If the scan chain passes all structural checks, the scan chain status will be "VALIDATED" (V). If the scan chain fails a structural check, the scan chain status will be "FAILED" (F).

If the **-file** option is not used, the **write_scan_def** command must first be issued to annotate the SCANDEF information to the current design database; otherwise this command issues an error message. See the **write_scan_def** man page for more information.

This command is used in the Design Compiler environment. In the IC Compiler environment, use the **check_scan_chain** command instead.

This command does not support scan chain validation against SCANDEF files written with the **write_scan_def -expand** command. When the **-expand** option is used, scan chain elements inside black-box CTL models are included in the SCANDEF file. This is needed when moving to IC Compiler, where the CTL models are replaced by the actual subdesigns. If the **check_scan_def** command used with such an expanded SCANDEF file, the report shows FAILED for all scan chains that have elements in black box subdesigns.

EXAMPLES

The following is an example of the output of the **check_scan_def** command when no file is specified:

```
prompt> check_scan_def
Checking SCANDEF...
  Checking scan cell correspondence between SCANDEF and netlist...
  Checking scan chain checking...
All checks completed.
```

```
Report : Scan DEF check
Design : design name
Version: software version
Date   : current date
```

```
Information from SCANDEF file:
Number of SCANCHAINS: 2
```

```
Checking between SCANDEF file and design:
Total SCANCHAINS checked: 2
VALIDATED : 2
FAILED   : 0
```

Chain name	Status	#cells	Scan IN	Scan OUT
1	V	62	U10/C	alu1/U381/A
2	V	61	U11/C	U9/B

1

The following example shows the result when a file is specified with the **check_scan_def** command:

```
prompt> check_scan_def -file temp.def
Reading DEF file temp.def
Checking SCANDEF...
  Checking scan cell correspondence between SCANDEF and netlist...
  Checking scan chain checking...
All checks completed.
```

```
Report : Scan DEF check
Design : design name
Version: software version
Date   : current date
```

```
Information from SCANDEF file:
Number of SCANCHAINS: 2
```

```
Checking between SCANDEF file and design:
```

Total SCANCHAINS checked: 2

VALIDATED : 2

FAILED : 0

Chain name	Status	#cells	Scan IN	Scan OUT
1	V	62	U10/C	alu1/U381/A
2	V	61	U11/C	U9/B

1

SEE ALSO

report_scan_chain(2)

write_scan_def(2)

check_scenarios

Performs consistency checks between the scenarios, for scenario specific information such as TLUplus files, operating conditions on the libraries, clocks and so on.

SYNTAX

```
status check_scenarios
      [-output ]
      [-display]
```

ARGUMENTS

-output

Specifies the directory to store the output. By default output is stored in HTML format in the directory, design_dir/check_scenarios_mmddyyyy_pid

-display

Opens a HTML browser to display the data. The browser to be used is defined by the Tcl variable, gui_online_browser.

DESCRIPTION

The **check_scenarios** command checks and reports issues with the scenarios. The command performs the following checks: TLUplus, clock related information such as clock periods, default versus nominal PVT on libraries, PVT from blocks versus parent, and PVT of library cells.

Multicorner-Multimode Support

This command uses information from all active scenarios. This command also supports designs that are not multicorner-multimode designs.

EXAMPLES

The following is an example of the **check_scenarios** command:

```
prompt> check_scenarios
check_scenarios
```

Warning: The complexity of the scenario (s1) might cause long runtime and high memory usage. (MCMM-212)

Warning: Scenario (s1) has instances which referenced library cells that are marked as "dont_use". (MCMM-225)

Warning: Scenario (s2) has a library setup issue with regard to the PVT values and the operating condition. (MCMM-213)

Information: HTML report can be found at

/WORKING_DIRECTORY/run/check_scenarios_7132009_21451/check_scenarios_0.html

SEE ALSO

gui_online_browser(3)

check_synlib

Performs semantic checks on synthetic libraries.

SYNTAX

status **check_synlib**

ARGUMENTS

This command has no arguments.

DESCRIPTION

The **check_synlib** command performs checking on synthetic libraries found in the synthetic library search path (**synthetic_library**) and listed by the **link_library** variable. Synthetic libraries can contain errors that cannot be detected using the **read_lib** command. Note that **check_synlib** also implicitly checks against standard.sldb.

The **check_synlib** command returns one of the following two values:

- 0 : Errors were reported.
- 1 : No critical errors were found.

Run **check_synlib** after you set the **synthetic_library** variable and the .sldb files exist, but before commands such as **compile** or **replace_synthetic** use any of these libraries.

EXAMPLES

The following example checks the synthetic libraries named in the **synthetic_library** variable for compatibility. The two libraries are my_synlib_0.sldb and my_synlib_1.sldb.

```
prompt> synthetic_library = {"my_synlib_0.sldb", "my_synlib_1.sldb"}
```

```
prompt> check_synlib
```


SEE ALSO

compile(2)
read_lib(2)
replace_synthetic(2)
report_synlib(2)
synthetic_library(3)

check_target_library_subset

Checks and prints out the inconsistent settings among target library, target library subset, and operating conditions.

SYNTAX

```
status check_target_library_subset
[-verbose]
[-max_messages inf]
```

ARGUMENTS

The **check_target_library_subset** command has no arguments.

DESCRIPTION

This command checks and prints out the inconsistent settings among target library, target library subset, and operating conditions. The command checks for the following:

- Conflicts between the target library subset and the global **target_library** variable. The target library subset should be a subset of the library set being specified by the **target_library** variable.
- Conflicts between operating condition and target library subset. There should be at least one library from the target library subset that has the same process, nom_voltage, and temperature as the operating condition being used on that block.
- Conflicts between the library cell of the mapped cell and target library subset. A warning is issued if the library cell of a mapped cell is not from any of the libraries of the target library subset. Note that this is not an error, because the subset applies only to new cells being created during optimization, not to existing cells in the block.

Multicorner-Multimode Support

Depending on the options used, this command either uses the current scenario or has no dependency on scenario-specific information.

SEE ALSO

check_library(2)
compile(2)

```
remove_target_library_subset(2)
report_target_library_subset(2)
set_operating_conditions(2)
set_target_library_subset(2)
target_library(3)
```

check_timing

Checks for possible timing problems in the current design.

SYNTAX

```
status check_timing
  [-overlap_tolerance minimum_distance]
  [-override_defaults check_list]
  [-multiple_clock]
  [-sdc_runtime]
  [-retain]
  [-include check_list]
  [-exclude check_list]
```

Data Types

<i>minimum_distance</i>	float
<i>check_list</i>	list

ARGUMENTS

-overlap_tolerance *minimum_distance*

Specifies the minimum distance allowed between the master-close edge and the slave-open edge. If the distance is less than the **-overlap_tolerance** value, the tool issues a warning message. Use this option to check for the master-slave clock overlap. By default, this option is off.

-override_defaults *check_list*

Overrides the checks defined by the **timing_check_defaults** variable by using the *check_list* argument. If you use the **-override_defaults** option with a *check_list*, you need to perform the final list of checks with the *check_list* argument of the **-override_defaults** option.

-multiple_clock

Issues a warning when multiple clocks reach a register clock pin. If more than one clock signal reaches a register clock pin, and the **timing_enable_multiple_clocks_per_reg** variable is set to **false**, the clock to use for analysis is undefined, and the tool issues a warning message. In this case, use either the **set_case_analysis** or **set_disable_timing** command so that only one clock can propagate from the sources to the register clock pin. By default, this option is off.

-sdc_runtime

Issues a report on the dirty constraints set on the design, which can contribute to high runtime. The default file name is **sdc_runtime.log**. Use the **sdc_runtime_analysis_log_file** variable to generate report with a different log file name. You can also generate this report by setting the **sdc_runtime_analysis_enable** variable to **true** and compile the design. When you use the **-**

sdcs_runtime option, the tool ignores all other options of the **check timing** command and generates only the SDC dirty constraint report. These checks are not influenced by the **timing_check_defaults** variable. By default, this option is off.

-retain

Issues a warning if any of the RETAIN values are larger than its corresponding delay value. The RETAIN values should be less than their corresponding delay values so that they can be considered for hold violations. Otherwise, the RETAIN values might be considered for setup violations. By default, this option is off.

-include check_list

Adds the checks listed in *check_list* to the checks defined by the **timing_check_defaults** variable.

-exclude check_list

Subtracts the checks listed in *check_list* from the checks defined by the **timing_check_defaults** variable.

DESCRIPTION

This command checks the timing attributes placed on the current design and issues warning messages as needed. The warning messages provide information that identifies and corrects potential errors. The warning messages do not necessarily indicate design problems.

This command without any options performs the checks defined by the **timing_check_defaults** variable. Redefine this variable to change the value. If the **-override_defaults** option is not used, the final list of checks to be performed is the list of checks specified by the **timing_check_defaults** variable, plus the list of checks specified by the **-include** option, minus the list of checks specified by the **-exclude** option. If you use the **-override_defaults** option with a *check_list*, you need to perform the final list of checks with the *check_list* argument of the **-override_defaults** option.

The alphabetically ordered list below shows the meaning of each check:

clock_crossing

Checks clock interactions when there are multiple clock domains. If a clock launches one or more paths that are captured by other clocks, it will have an entry in the clock crossing report. If all paths between two clocks are false paths or they are exclusive or asynchronous clocks, the path is marked with an asterisk (*). If only some of the paths are set as false paths, the path is marked by a number sign (#).

clock_no_period

Warns if clocks with no period are specified.

data_check_multiple_clock

Warns if multiple clocked signals reach a data check register reference pin. The analysis is done separately for each of the clocked domains. If the **timing_enable_multiple_clock_per_reg** variable is set to **false**, only one of the clocked signals is analyzed.

data_check_no_clock

Warns if no clocked signal reaches a data check register reference pin. In this case, no setup or hold checks are performed on the constrained pin.

gated_clock

Warns about the gated clocks. Disables the gating timing arcs only if the **propagated_clock** attribute is set on that clock.

generated_clock

Checks the generated clock network. The following types of issues are reported:

- The source pin (master point) is not the clock source.
- The definition point of the generated clock has no path to the sourcepoint.
- The generated clocks form a loop.

generic

Warns about generic (unmapped) cells in the design. The timing of paths through generic cells is inaccurate, because the generic cells have zero delay.

ideal_clocks

Warns about any clock networks that are ideal. Generally, all clocks should be propagated so that the clock network timing is accurately calculated. Especially in the presence of crosstalk, the delay changes induced by other nets on the clock network are not reflected in the calculated slacks in the design.

ideal_timing

Warns when the user-defined ideal transition or latency is set on a normal pin (not ideal).

loops

Warns about combinational feedback loops. If the feedback loop is not broken by the **set_disable_timing** command, it is automatically broken by disabling one or more timing arcs.

multiple_clock

This item should be specified with the **-multiple_clock** option.

net_no_driving_info

Warns about any net that does not have driving pins or that there are no timing arcs on the driver pins. The warning is issued only when the net has coupled parasitic.

no_driving_cell

Warns about any port that does not have a driving cell. The warning is issued only when the net connected to the port has parasitic. When no driving cell is specified, the net is assigned to a strong driver for modeling aggressor effects, which can be pessimistic. Also, a port with no driving cell could act as a strong victim, which could underestimate the crosstalk effect.

no_input_delay

Warns if no clock-related delay is specified on an input port, where it propagates to a clocked latch or output port. If the **timing_input_port_default_clock** variable is set to **true**, a default clock is assumed for the input port. Otherwise, it will not be clocked, and the paths are unconstrained. In this case, if there is no input delay specified, the **check_timing** command does not generate warnings.

partial_input_delay

Warns about any ports have partially defined input delay; only the minimum or only the maximum delay defined with the **set_input_delay** command. As a result, some paths starting from the port with partially defined input delay might become unconstrained and some potential violations could be missed.

pulse_clock_cell_type

Warns if an instance cell has a mismatched pulse type defined from its library cell.

retain

This check item should also be specified by the **-retain** option.

unconstrained_endpoints

Warns about unconstrained timing endpoints. This warning identifies timing endpoints that are not constrained for maximum delay (setup) checks. If the paths to the endpoint are all false paths, endpoints are not reported as unconstrained endpoints.

The warning messages that can occur when you use this command are described below.

- The following warning message is issued when the waveforms applied to a master-slave register are overlapping or have different periods. Use the **create_clock** command to modify one of the waveforms.

The overlap check first determines the intended master-slave waveform relationships based on the ideal clock waveforms. Then, the clock network delay and uncertainty is applied to the waveforms and if the distance between the related edges is less than the overlap tolerance, the tool issues a warning message.

WARNING: The following master-slave registers have overlapping clock violations. The waveforms or periods might be invalid.

- The following warning message identifies timing endpoints (output ports and register data pins) that are not constrained. If the endpoint is a register data pin, use the **create_clock** command for the appropriate clock source to constrain the pin. Use the **set_output_delay** or **set_max_delay** command to constrain output ports. The **set_output_delay** command constrains only the path when the delay is relative to a clock.

WARNING: The following endpoints are not constrained for maximum delay.

- The following warning message identifies gated clocks. Disable the gating timing arcs only if the **propagated_clock** attribute is set on that clock.

WARNING: The clock network starting at "clk" is gated by the following input pins. The gating timing arcs might need to be disabled for clocks with the "propagated_clock" attribute.

- The following warning message is issued when the design contains unmapped cells (generic logic), which causes the timing of the paths through the generic cells to be inaccurate because generic cells have zero delay.

WARNING: Design "*design_name*" contains unmapped cells.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example checks the timing of the current design and issues warning messages as needed:

```
prompt> check_timing
Information: Checking 'unexpandable_clocks'...
Information: Checking 'generic'...
Information: Checking 'latch_fanout'...
Information: Checking 'loops'...
Information: Checking 'generated_clocks'...
```

SEE ALSO

- check_design(2)
- check_library(2)
- compile(2)
- compile_ultra(2)
- create_clock(2)
- current_design(2)
- set_case_analysis(2)
- set_disable_timing(2)
- set_max_delay(2)
- set_output_delay(2)
- timing_enable_multiple_clocks_per_reg(3)

check_tlu_plus_files

Checks the files used for TLUPlus extraction.

SYNTAX

status **check_tlu_plus_files**

ARGUMENTS

This command has no arguments.

DESCRIPTION

The **check_tlu_plus_files** command performs consistency checks on the TLUPlus files used for virtual route and postroute extraction, which are specified by using the **set_tlu_plus_files** command. If the TLUPlus files are from a Milkyway design library attachment, this command does not do any checking because the attachment was already checked when it was attached.

The following items are checked for consistency:

- Layer names
The consistency check ensures that the conductor and via layer names are consistent across the Milkyway, ITF, and mapping files. The tool also checks that the number of layers are consistent between the ITF files and the Milkyway technology files. The tool issues error messages if there are inconsistencies.
- Etch values
The tool checks that the etch values in the ITF files are consistent with the delta width values in the Milkyway technology files under minimum, nominal, and maximum conditions. The signs used to indicate expand and shrink are opposite in the ITF files and the Milkyway technology files. The tool issues warning messages if there are inconsistencies.
- Minimum width and minimum spacing
The tool checks that the minimum width and minimum spacing are consistent between the ITF files and the Milkyway technology files, and among different minimum, nominal, and maximum ITF files. The tool issues warning messages if there are inconsistencies.
- Conductor thickness
The tool checks that the conductor thicknesses are consistent between the ITF files and the Milkyway technology files. The tool issues warning messages if there are inconsistencies.

Multicorner-Multimode Support

This command uses information from all scenarios.

EXAMPLES

The following example runs the consistency checks:

```
prompt> check_tlu_plus_files
```

SEE ALSO

check_library(2)
set_tlu_plus_files(2)

check_upf

Checks the UPF loaded on a hierarchical design for validity of references across the abstract boundary between the top level and the block level.

SYNTAX

status **check_upf**

ARGUMENTS

This command has no arguments.

DESCRIPTION

In hierarchical flow designs, the **save_upf** command saves only the UPF constraints that apply to the top-level design. This command tests the UPF loaded into the current design for any reference that crosses the boundary between the top level and the block level, and generates a warning for each UPF command or argument that will be dropped when the **save_upf** command is used.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> check_upf
Warning: The following elements have been dropped from a UPF statement: mid/block (UPF-701)
Warning: UPF statement 'set_isolation' will be dropped because its scope belongs to an
abstract hierarchy. (UPF-702)
0
```

SEE ALSO

load_upf(2)
save_upf(2)

clean_buffer_tree

Removes the buffer tree at the specified driver pin, load pin, or driver net on a mapped design.

SYNTAX

```
status clean_buffer_tree
  [-from start_point_list]
  [-to object_list]
  [-net net_list]
  [-source_of object_list]
  [-hierarchy]
  [-global]
  [-threshold integer_in_1]
```

Data Types

```
start_point_list  list
object_list       list
net_list          list
integer_in_1     integer
```

ARGUMENTS

-from *start_point_list*

Specifies the starting point of the buffer tree to be removed. The *start_point_list* must contain only pins or ports. This option can be used in conjunction with the **-to** and **-net** options.

-to *object_list*

The buffer tree removal starts at the immediate buffer source of the specified pins or ports. The *object_list* must contain only pins or ports. This option can be used in conjunction with the **-from** and **-net** options.

-net *net_list*

Specifies the driver net that is the starting point for the buffer tree to be removed. The *net_list* must contain only nets. This option can be used in conjunction with the **-from** and **-to** options.

-source_of *object_list*

The complete buffer tree in the fanin to the object up to the object itself is removed. The *object_list* must contain only output ports or input pins. This option cannot be used in conjunction with the **-from**, **-to** or **-net** options.

-hierarchy

Removes the buffer tree across the hierarchy. By default, the buffer tree removal stops when it reaches the hierarchy boundary.

-global

Searches for high-fanout buffer tree root pins in the *net_list*. This option is not valid with the **-from**, **-to**, and **-net** options.

-threshold *integer_in_1*

Specifies the fanout threshold range that determines which buffer trees are cleaned up. If the number of sink pins driven from the primary root pin of the buffer tree is larger than the threshold, the buffer tree circuitry is cleaned up. This option is only valid with the **-global** option.

Valid **-threshold** values are from 1 to 1,000,000.

DESCRIPTION

The **clean_buffer_tree** command removes a buffer tree at a specified driver pin, load pin, or driver net on a mapped design. When given a driver pin or driver net, the buffer tree is removed with this driver as the root. When given a load pin, the buffer tree is removed with the immediate driver of the load pin as the root. By default, this command does not remove the buffer tree across the hierarchy. To enable buffer tree removal across boundaries, specify the **-hierarchy** option when running the command.

The buffer tree removal process starts at the source and removes all buffer and inverter cells in its transitive fanout cone until it finds nonbuffer cells or hierarchy boundaries. This does not apply if the **-hierarchy** option is specified.

Buffer and inverter cells that have the **dont_touch** attribute or that are connected to a net that has the **dont_touch** attribute are not removed. Similarly, any cells that follow these cells in the transitive fanout are not removed.

The **clean_buffer_tree** command adds an inverter driving all of the original buffer tree's inverted loads after it has removed all buffers and inverters in the tree.

This command accepts the final implementation of buffer tree removal, even if it negatively impacts the timing or DRC violations in the design.

EXAMPLES

The following example removes a buffer tree without crossing hierarchical boundaries:

```
prompt> clean_buffer_tree -from cell1/O
```

The following example removes a buffer tree that crosses hierarchical boundaries:

```
prompt> clean_buffer_tree -hierarchy -from cell1/O
```

SEE ALSO

- all_fanout(2)
- balance_buffer(2)
- report_buffer_tree(2)
- report_cell(2)
- report_constraint(2)

close_mw_lib

Closes the current Milkyway library.

SYNTAX

status **close_mw_lib**

ARGUMENTS

The **close_mw_lib** command has no arguments.

DESCRIPTION

This command closes the current Milkyway library. It returns a status indicating success or failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example closes the current Milkyway library:

```
prompt> close_mw_lib  
1
```

SEE ALSO

open_mw_lib(2)

collections

Describes the methodology for creating collections of objects and querying objects in the database.

DESCRIPTION

Synopsys applications build an internal database of objects and attributes applied to them. These databases consist of several classes of objects, including designs, libraries, ports, cells, nets, pins, clocks, and so on. Most commands operate on these objects.

By definition:

A collection is a group of objects exported to the Tcl user interface.

Collections have an internal representation (the objects) and, sometimes, a string representation. The string representation is generally used only for error messages.

Collections represent an ordered sequence of database objects. These collections provide constant time random access to the objects contained in the sequence.

A set of commands to create and manipulate collections is provided as an integral part of the user interface. The collection commands encompass two categories: those that create collections of objects for use by another command, and one that queries objects for viewing. The result of a command that creates a collection is a Tcl object that can be passed along to another command. For a query command, although the visible output looks like a list of objects (a list of object names is displayed), the result is an empty string.

An empty string "" is equivalent to the empty collection, that is, a collection with zero elements.

To illustrate the usage of the common collection commands, the man pages have examples. Most of the examples use PrimeTime as the application. In all cases, the application from which the example is derived is indicated.

Homogeneous and Heterogeneous Collections

A homogeneous collection contains only one type of object. A heterogeneous collection can contain more than one type of object. Commands that accept collections as arguments can accept either type of collection.

Lifetime of a Collection

Collections are active only as long as they are referenced. Typically, a collection is referenced when a variable is set to the result of a command that creates it or when it is passed as an argument to a command or a procedure. For example, in PrimeTime, you can save a collection of design ports by setting a variable to the result of the **get_ports** command:

```
pt_shell> set ports [get_ports *]
```

Next, either of the following two commands deletes the collection referenced by the *ports* variable:

```
pt_shell> unset ports
pt_shell> set ports "value"
```

Collections can be implicitly deleted when they go out of scope. Collections go out of scope for various reasons. An example would be when the parent (or other antecedent) of the objects within the collection is deleted. For example, if our collection of ports is owned by a design, it is implicitly deleted when the design that owns the ports is deleted. When a collection is implicitly

deleted, the variable that referenced the collection still holds a string representation of the collection. However, this value is useless because the collection is gone, as illustrated in the following PrimeTime example:

```
pt_shell> current_design
{"TOP"}

pt_shell> set_ports [get_ports in*]
{"in0", "in1"}

pt_shell> remove_design TOP
Removing design 'TOP'...

pt_shell> query_objects $ports
Error: No such collection '_sel26' (SEL-001)
```

Iteration

To iterate over the objects in a collection, use the **foreach_in_collection** command. You cannot use the Tcl-supplied **foreach** iterator to iterate over the objects in a collection, because the **foreach** command requires a list, and a collection is not a list. In fact, if you use the **foreach** command on a collection, it destroys the collection.

The arguments of the **foreach_in_collection** command are similar to those of **foreach**: an iterator variable, the collection over which to iterate, and the script to apply at each iteration. Note that unlike the **foreach** command, the **foreach_in_collection** command does not accept a list of iterator variables.

The following example is an iterative way to perform a query in PrimeTime. For more information, see the **foreach_in_collection** man page.

```
pt_shell> \
foreach_in_collection s1 $collection {
  echo [get_object_name $s1]
}
```

Manipulating Collections

A variety of commands are provided to manipulate collections. In some cases, a particular command might not operate on a collection of a specific type. This is application-specific. Consult the man pages from your application.

- **add_to_collection** - This command creates a new collection by adding a list of element names or collections to a base collection. The base collection can be the empty collection. The result is a new collection. In addition, the **add_to_collection** command allows you to remove duplicate objects from the collection by using the *-unique* option.
- **append_to_collection** - This command appends a set of objects (specified by name or collection) to an existing collection. The base collection is passed in through a variable name, and the base collection is modified directly. It is similar in function to the **add_to_collection** command, except that it modifies the collection in place; therefore, it is much faster than the **add_to_collection** command when appending.
- **remove_from_collection** - This command removes a list of element names or collections from an existing collection. The second argument is the specification of the objects to remove and the first argument is the collection to have them removed from. The result of the command is a new collection. For example, in PrimeTime:

```
pt_shell> set dports \
[remove_from_collection [all_inputs] CLK]
{"in1", "in2", "in3"}
```

- **compare_collections** - This command verifies that two collections contain the same objects (optionally, in the same order). The result is "0" on success.
- **copy_collection** - This command creates a new collection containing the same objects in the same order as a given

collection. Not all collections can be copied.

- **index_collection** - This command extracts a single object from a collection and creates a new collection containing that object. The index operation is done in constant time - it is independent of the number of elements in the collection, or the specific index. Not all collections can be indexed.
- **sizeof_collection** - This command returns the number of objects in a collection.

Filtering

In some applications, you can filter any collection by using the **filter_collection** command. This command takes a base collection and creates a new collection that includes only those objects that match an expression.

Some applications provide a *-filter* option for their commands that create collections. This allows objects to be filtered out before they are ever included in the collection. Frequently this is more efficient than filtering after they are included in the collection. The following examples from PrimeTime filters out all leaf cells:

```
pt_shell> filter_collection \
[get_cells *] "is_hierarchical == true"
{"i1", "i2"}
pt_shell> get_cells * -filter "is_hierarchical == true"
{"i1", "i2"}
```

The basic form of a filter expression is a series of relations joined together with AND and OR operators. Parentheses are also supported. The basic relation contrasts an attribute name with a value through a relational operator. In the previous example, *is_hierarchical* is the attribute, *==* is the relational operator, and *true* is the value.

The relational operators are

```
== Equal
!= Not equal
> Greater than
< Less than
>= Greater than or equal to
<= Less than or equal to
=~ Matches pattern
!~ Does not match pattern
```

The basic relational rules are

- String attributes can be compared with any operator.
- Numeric attributes cannot be compared with pattern match operators.
- Boolean attributes can be compared only with *==* and *!=*. The value can be only true or false.

Additionally, existence relations determine if an attribute is defined or not defined, for the object. For example,

```
(sense == setup_clk_rise) and defined(sdf_cond)
```

The existence operators are

```
defined
undefined
```

These operators apply to any attribute as long as it is valid for the object class. See the appropriate man pages for complete details.

Sorting Collections

In some applications, you can sort a collection by using the **sort_collection** command. It takes a base collection and a list of attributes as sort keys. The result is a copy of the base collection sorted by the given keys. Sorting is ascending, by default, or descending when you specify the *-descending* option. In the following example from PrimeTime, the command sorts the ports by direction and then by full name.

```
pt_shell> sort_collection [get_ports *] \
{direction full_name}
{"in1", "in2", "out1", "out2"}
```

Implicit Query of Collections

In many applications, commands that create collections implicitly query the collection when the command is used at the command line. Consider the following examples from PrimeTime:

```
pt_shell> set_input_delay 3.0 [get_ports in*]
1
pt_shell> get_ports in*
{"in0", "in1", "in2"}
pt_shell> query_objects -verbose [get_ports in*]
{"port:in0", "port:in1", "port:in2"}
pt_shell> set_iports [get_ports in*]
{"in0", "in1", "in2"}
```

In the first example, the **get_ports** command creates a collection of ports that is passed to the **set_input_delay** command. This collection is not the result of the primary command (**set_input_delay**), and as soon as the primary command completes, the collection is destroyed. The second example shows how a command that creates a collection automatically queries the collection when that command is used as a primary command. The third example shows the verbose feature of the **query_objects** command, which is not available with an implicit query. Finally, the fourth example sets the variable **iports** to the result of the **get_ports** command. Only in the final example does the collection persist to future commands until **iports** is overwritten, unset, or goes out of scope.

SEE ALSO

```
add_to_collection(2)
as_collection(2)
append_to_collection(2)
compare_collections(2)
copy_collection(2)
filter_collection(2)
foreach_in_collection(2)
index_collection(2)
query_objects(2)
remove_from_collection(2)
sizeof_collection(2)
sort_collection(2)
```

compare_collections

Compares the contents of two collections. If the same objects are in both collections, the result is "0" (like string compare). If they are different, the result is nonzero. The order of the objects can optionally be considered.

SYNTAX

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

Data Types

<i>collection1</i>	collection
<i>collection2</i>	collection

ARGUMENTS

-order_dependent

Indicates that the order of the objects is to be considered; that is, the collections are considered to be different if the objects are ordered differently.

collection1

Specifies the base collection for the comparison. The empty string (the empty collection) is a legal value for the *collection1* argument.

collection2

Specifies the collection with which to compare to *collection1*. The empty string (the empty collection) is a legal value for the *collection2* argument.

DESCRIPTION

The **compare_collections** command is used to compare the contents of two collections. By default, the order of the objects does not matter, so that a collection of cells u1 and u2 is the same as a collection of the cells u2 and u1. By using the **-order_dependent** option, the order of the objects is considered.

Either or both of the collections can be the empty string (the empty collection). If two empty collections are compared, the comparison succeeds (that is, **compare_collections** considers them identical), and the result is "0".

EXAMPLES

The following example from PrimeTime shows a variety of comparisons. Note that a result of "0" from **compare_collections** indicates success. Any other result indicates failure.

```
prompt> compare_collections [get_cells *] [get_cells *]
0
prompt> set c1 [get_cells {u1 u2}]
{u1 u2}
prompt> set c2 [get_cells {u2 u1}]
{u2 u1}
prompt> set c3 [get_cells {u2 u4 u6}]
{u2 u4 u6}
prompt> compare_collections $c1 $c2
0
prompt> compare_collections $c1 $c2 -order_dependent
-1
prompt> compare_collections $c1 $c3
-1
```

The following example builds on the previous example by showing how empty collections are compared.

```
prompt> set c4 ""
prompt> compare_collections $c1 $c4
-1
prompt> compare_collections $c4 $c4
0
```

SEE ALSO

[collections\(2\)](#)

compare_delay_calculation

Compares the Arnoldi-based delays with the Elmore delays in the current design.

SYNTAX

```
status compare_delay_calculation  
  [-verbose]  
  [-ccs]
```

ARGUMENTS

-verbose

Creates histogram-style reports showing the distribution of the delay differences when using the Arnoldi-based and Elmore delay models.

-ccs

Creates histogram-style reports showing the delay differences when using Composite Current Source (CCS) models and nonlinear delay models (NLDLM).

DESCRIPTION

The **compare_delay_calculation** command calculates and compares the timing results of the current design using the Arnoldi-based and Elmore delay models. This command reports pin-to-pin delays, max transition, several DRC violations and the worst slack within each path group for both delay models. In the verbose mode this command creates histogram-style reports showing the distribution of the Arnoldi-based and Elmore delay differences on the paths with negative slacks. The distribution of the differences is represented by the two histograms showing the delay differences in the library units and in the percentage.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following is an example of an Arnoldi-based and Elmore delay calculation report:

```
prompt> compare_delay_calculation -verbose
```

Percent of Arnoldi-based delays = 66.67%

+=====+				
	Elmore		Arnoldi	
+-----+				
Max Transition	0.195		0.521	
DRC Violations	1		1	
+-----+				
Group	WNS	TNS	WNS	TNS
+-----+				
CLK1	1.71	3.96	1.46	2.64
default	0.00	0.00	0.00	0.00
+-----+				

Histograms for Delays with Negative Slack

+=====+			
Difference (%)	Count	%	
+-----+			
0 - 1	2	8.33	
1 - 5	5	20.83	
5 - 10	3	12.50	
10 - 50	10	41.67	
50 - 100	4	16.67	
100+	0	0.00	
+=====+			
Difference(lib_units)	Count	%	
+-----+			
0.0 - 0.001	0	0.00	
0.001 - 0.01	3	12.50	
0.01 - 0.05	7	29.17	
0.05 - 0.1	0	0.00	
0.1 - 1.0	14	58.33	
1.0+	0	0.00	
+=====+			

Histograms for Slews with Negative Slack

+=====+			
Difference (%)	Count	%	
+-----+			
0 - 1	0	0.00	
1 - 5	8	33.33	
5 - 10	9	37.50	
10 - 50	3	12.50	
50 - 100	4	16.67	
100+	0	0.00	
+=====+			
Difference(lib_units)	Count	%	
+-----+			
0.0 - 0.001	0	0.00	
0.001 - 0.01	12	50.00	
0.01 - 0.05	0	0.00	
0.05 - 0.1	0	0.00	
0.1 - 1.0	12	50.00	
+=====+			

1.0+		0		0.00	
+-----+					

SEE ALSO

- read_parasitics(2)
- report_delay_calculation(2)
- report_timing(2)

compare_lib

Performs a cross-reference check between a technology library and a symbol library or between a technology library and a physical library.

SYNTAX

```
status compare_lib  
      library1  
      library2
```

Data Types

```
library1  string  
library2  string
```

ARGUMENTS

library1

Specifies the name of a technology library.

library2

Specifies the name of a symbol library or physical library to be compared with the technology library.

DESCRIPTION

The **compare_lib** command compares the specified libraries consistency. The first library is a technology library and the second library is a symbol or physical library. For this command to run successfully, the libraries to be compared must be in the Synopsys internal database format and must also exist in memory. To compile a library, use the **read_lib** command. To load a compiled library, use the **read_file** command.

The **compare_lib** command performs the following checks:

- First, it ensures that each component in the technology library has a corresponding definition in the symbol or physical library.
- Second, it crosschecks the pin names of each component in the technology library against the pin names defined for its corresponding symbol or physical representation.
- Third, it ensures that each component in the symbol or physical library has a corresponding definition in the technology library.
- Fourth, it crosschecks the pin names of each component in the symbol or physical library against the pin names defined for its

corresponding technology cell.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example compares a technology library and a symbol library:

```
prompt> compare_lib tech_lib.db sym_lib.sdb
```

SEE ALSO

read_file(2)
read_lib(2)
report_lib(2)
write_lib(2)

compare_supplies

compares voltage levels or relative ON-ness between two given supplies.

SYNTAX

```
status compare_supplies  
  supply_list  
  [-on_status]  
  [-voltage_level]
```

Data Types

supply_list list

ARGUMENTS

supply_list

Specifies a list of 2 supplies. The first supply specified is treated as the 'reference' supply to which the second supply ('specified' supply) will be compared to. Supplies can either be supply net names or supply set handles.

-on_status

If specified, this option will return one of 4 values: "equal" if the reference supply is equal to the specified supply. "more_on" if the specified supply is more ON as compared to the reference supply. "less_on" if the specified supply is less ON as compared to the reference supply. "independent" if the specified supply may be more or less ON as compared to the reference supply depending on the PST State.

-voltage_level

If specified, this option will return one of 4 values: "equal", if both reference and specified supplies are at the same voltage and no level-shifting is needed. "higher", if reference supply is lower voltage as compared to the specified supply. "lower", if the reference supply is higher voltage as compared to the specified supply. "independent", if the reference supply is higher/lower voltage as compared to the specified supply depending on the PST State. If the two supplies specified belong to the same correlated supply group, they will be compared based on the rules for supplies in the same correlated supply group.

DESCRIPTION

This command is used to compare voltage levels or relative always-ON levels of two given supplies. Options -on_status and -voltage_level can be used separately or in combination. Supplies list must consist of exactly 2 supplies.

SEE ALSO

report_pst(2)

compile

Performs logic-level and gate-level synthesis and optimization on the current design.

SYNTAX

```
status compile
  [-no_map]
  [-map_effort medium | high]
  [-area_effort none | low | medium | high]
  [-incremental_mapping]
  [-exact_map]
  [-ungroup_all]
  [-boundary_optimization]
  [-auto_ungroup area | delay]
  [-no_design_rule
   | -only_design_rule
   | -only_hold_time]
  [-scan]
  [-top]
  [-power_effort none | low | medium | high]
  [-gate_clock]
```

ARGUMENTS

-no_map

Specifies not to map the current design into the target technology library. With this option, generic Boolean equations and generic flip-flops represent the resulting design.

-map_effort medium | high

Specifies the relative amount of CPU time spent during the mapping phase of **compile**. Valid values are **medium** and **high**. You can select one value. The default is **medium**.

-area_effort none | low | medium | high

Specifies the relative amount of CPU time spent during the area recovery phase of **compile**. Valid values are **none**, **low**, **medium**, and **high**. You can select one value. The default is the specified *map_effort*.

-incremental_mapping

Specifies to attempt only incremental improvements to the gate structure of a design. Portions of a design that are already mapped are exempt from logic-level optimization, and the resulting design should be the same (if no improvements can be made) or better in terms of its design constraints. Implementations for DesignWare operators are reselected in an incremental compile run if the swap can improve the optimization cost based on the optimization constraints on the design.

-exact_map

Specifies that the sequential elements in the final design must exactly match the descriptions specified in the HDL description. SEQGEN is a technology-independent representation of a sequential element that is inferred by HDL Compiler from HDL descriptions of sequential circuits. When you specify this option, the tool attempts to find sequential elements in the target technology library that match the behavior of only the SEQGEN element (that is, no surrounding logic is considered). In the event that an exact match for a SEQGEN is not available, a different sequential element is used. Use of the **-exact_map** option would disable sequential output inversion. For more information, see **compile_seqmap_enable_output_inversion**. For improved sequential inference, use this option in conjunction with the HDL directives **sync_set_reset**, **sync_set_reset_local**, and **sync_set_reset_all**. The new attributes or directives tell HDL Compiler how to connect nets to the SEQGEN component. Use of the attributes and directives with the **-exact_map** option ensures that the results of **compile** are predictable. Use of the **-exact_map** option does not mean the QN pin won't be used in the mapped sequential element.

-ungroup_all

Collapses all levels of hierarchy in a design, except those that have the **dont_touch** attribute set.

-boundary_optimization

Optimizes across all hierarchical boundaries in the design. This can change the function of the design so that it can operate only in its current environment. If input or output ports are complemented as a result of using this option, port names are changed according to the **port_complement_naming_style** variable.

-auto_ungroup area | delay

Specifies to automatically ungroup hierarchies in the design. A hierarchy is considered for automatic ungrouping only if it satisfies all of the following criteria:

- It does not have the **dont_touch** or **ungroup** attribute set on it.
- There are no timing constraints or exceptions set on its pins.
- Its wire load model is the same as that of its parent, or the **compile_auto_ungroup_override_wlm** variable is set to **true**.
- It has fewer child cells than the value of the **compile_auto_ungroup_area_num_cells** variable if you select **area**.

If you select **area**, all hierarchies that meet the above criteria are ungrouped. The **-auto_ungroup area** option is generally used to ungroup small hierarchies in the design to improve area recovery. It does not have a significant impact on the timing of the design.

The **-auto_ungroup delay** option employs an intelligent ungrouping strategy that attempts to improve the overall timing of the design. It chooses hierarchies that are most likely to benefit from the extra boundary optimizations that ungrouping exposes. The algorithm emphasizes hierarchies containing paths that are either critical, or likely to become critical, after subsequent optimization steps.

Delay-driven auto-ungrouping offers a less CPU-intensive alternative to using the **-ungroup_all** option for improving design timing.

-no_design_rule

Determines, along with the **-only_design_rule** and **-only_hold_time** options, whether the command fixes design-rule violations before exiting. The **-no_design_rule** option specifies for the command to exit before fixing design-rule violations, thus allowing you to check the results in a constraint report before fixing the violations. The **-no_design_rule**, **-only_design_rule**, and **-only_hold_time** options are mutually exclusive. You can use only one of the options. The default is to perform both design-rule fixing and mapping optimizations before exiting.

-only_design_rule

Determines, along with the **-no_design_rule** and **-only_hold_time** options, whether the command fixes design-rule violations before exiting. The **-only_design_rule** option specifies for the command to perform only design-rule fixing; that is, mapping optimizations are not performed. The **-no_design_rule**, **-only_design_rule**, and **-only_hold_time** options are mutually exclusive.

You can use only one option. The default is to perform both design-rule fixing and mapping optimizations before exiting.

-only_hold_time

Determines, along with the **-no_design_rule** and **-only_design_rule** options, whether the command fixes design-rule violations before exiting. The **-only_hold_time** option specifies for the command to perform only hold-time fixing, ignoring other design rules. The **set_fix_hold** command must be specified for hold-time fixing to be performed. The **-no_design_rule**, **-only_design_rule**, and **-only_hold_time** options are mutually exclusive. You can select only one option. The default is to perform both design-rule fixing and mapping optimizations before exiting.

-scan

Specifies that the command is to consider the impact of scan insertion on mission-mode constraints during optimization. This option causes the command to replace all sequential elements during optimization. Some scan-replaced sequential cells might be converted to nonscan cells later in the test synthesis process because of test design-rule violations or explicit user specifications. By accounting for the impact of internal scan insertion from the start of the design process, test-ready compile eliminates the need for future reoptimization.

-top

Fixes design-rule and top-level timing violations for a design but does not perform any mapping on the design. By default, this option fixes all design-rule violations, but it only fixes the timing violations whose paths cross top-level hierarchical boundaries. If you want this option to fix timing violations for all paths, set the **compile_top_all_paths** variable to **true**.

-power_effort none | low | medium | high

Specifies the relative amount of CPU time spent during the power optimization phase of **compile**. Valid values are **none**, **low**, **medium**, and **high**. You can select one value. The default is the specified *map_effort*. Since power optimization in **compile** is enabled by power constraints, this option is ignored if there is no power constraint on the design.

-gate_clock

Enables clock gating optimization: clock gates are automatically inserted or removed. The **-gate_clock** option cannot be used in combination with the **-only_design_rule** option. When used in combination with the **-exact_map** option, it may not be possible to honor the **-exact_map** option for those registers that are involved with clock-gating optimization.

A clock-gating cell is not modified or removed if it or its parent hierarchical cell is marked **dont_touch** with the **set_dont_touch** command.

DESCRIPTION

The **compile** command performs logic-level and gate-level synthesis and optimization on the current design. Optimization is controlled by user-specified constraints on the design. These constraints describe goals for the optimization process, such as making the smallest circuit possible or trying to make specified outputs arrive by a specified time. The optimization process trades off timing and area constraints to provide the smallest possible circuit that meets specified timing requirements. Values for the area and speed of components used in synthesizing and optimizing the design are obtained from user-specified libraries.

Constraints fall into one of two categories: design rule constraints and optimization constraints. Design rule constraints reflect technology-specific restrictions that must be met for a design to function correctly. Optimization constraints reflect less critical design goals and restrictions that are desirable but not crucial for the operation of a design.

The design rule constraints are **max_transition**, **max_fanout**, **max_capacitance**, and **min_capacitance**. All other constraints are optimization constraints. Examples include maximum delay and maximum area.

During optimization, both types of constraints are considered; however, precedence is given to meeting design rule constraints. The **min_delay** and **fix_hold** constraints are treated similarly to design rule constraints, except that they have lower priority than the

maximum delay constraints. The priority given to various constraints can be modified by using the **set_cost_priority** command.

Often when a design read from an HDL description is optimized, new designs modeled from the synthetic library are generated. These designs are named according to the style specified in the **synthetic_design_naming_style** variable.

When the current design is hierarchical, and there are multiple design instances of a subdesign, it is not necessary to run the **uniquify** command before running the **compile** command. The **uniquify** command can still be used on multiple design instances so they can be individually optimized. For all designs marked with the **uniquify** attribute, **compile** copies and renames them according to the **uniquify_naming_style** variable.

The **compile** command does not optimize across hierarchical boundaries unless the **boundary_optimization** attribute is set to **true**. All synthetically generated designs are created with this attribute set to **true**. If boundary optimization causes ports to be complemented, port names are changed according to the **port_complement_naming_style** variable.

To customize the way the **compile** command optimizes subdesigns, use compile directives. Commands for placing these directives include **set_flatten** and **set_structure**. Command-line options pass global directives that affect how the entire design is optimized to the **compile** command.

If the current design or any of its subdesigns is represented as a state table, the **compile** command automatically performs finite state machine optimizations on these designs. Finite state machine synthesis is also controlled with compile directives placed directly on these designs by using commands such as **set_fsm_encoding** and **set_fsm_minimize**.

If the current design or any of its subdesigns contains arithmetic operations (additions, subtractions, multiplications, and comparisons), the **compile** command automatically transforms them into datapath blocks to be implemented by a datapath generator. Datapath optimization can be controlled by using the **hlo_disable_datapath_optimization** variable.

The **compile** command reports progress in real time by displaying a report. During optimization, the report shows incremental results each time a transformation is applied to the design. The default fields of the report are ELAPSED TIME, AREA, WORST NEG SLACK, TOTAL NEG SLACK, DESIGN RULE COST, and ENDPOINT.

ELAPSED TIME

Tracks the elapsed time since the beginning of the current **compile** or **reoptimize_design**.

AREA

Shows the area of the design during the optimization.

WORST NEG SLACK

Shows the worst negative slack (max_path violation) in all path groups.

TOTAL NEG

Shows the sum of the negative slack across all endpoints in the design.

DESIGN RULE COST

Measures the distance between the actual results and the user-specified design rule constraints.

ENDPOINT

Shows the endpoint currently being worked on. When a delay violation is being fixed, the object for the ENDPOINT is a pin or a port. When a design rule violation is being fixed, the object for the ENDPOINT is a net.

You can specify other fields in addition to or instead of any of the default fields. For a list of available fields and other details about specifying the compile log format, see the **compile_log_format** variable man page.

To display the same log format as the 1998.02 version, set **compile_log_format** = "". The fields for the 1998.02 version are TRIALS, AREA, DELTA DELAY, TOTAL NEGATIVE SLACK, and DESIGN RULE COST. The new default does not have the TRIALS and DELTA DELAY fields.

TRIALS

Tracks the number of transformations that the optimizer tries before making the current selection.

DELTA DELAY

Shows the current maximum delay cost of the design, which is the sum of the worst negative slack (max_path violation) in each path group.

The log written by **compile** shows the different phases of optimization. In addition to the Resource Allocation and Mapping phases, there are four more phases. The first of these is the Delay Optimization phase, which fixes max_path violations. In the Phase 1 Design Rule Fixing phase, design-rule violations are fixed without creating any new max_path delay violations. In the Phase 2 Design Rule Fixing phase, max_path delay constraints might be impacted while fixing design-rule violations. Finally, there is an Area Recovery phase that attempts to reduce the area of the design while keeping other constraints intact. The Area Recovery phase always does some minimal cleanup, but it performs more CPU-intensive area recovery only if max_area constraints have been set on the design. If the **-only_design_rule** option is used, the Delay Optimization phase and Area Recovery phase are skipped. If the **-no_design_rule** option is used, both Phase 1 Design Rule Fixing and Phase 2 Design Rule Fixing are skipped. If the **set_cost_priority** command with the **-delay** option has been used before using the **compile** command, Phase 2 Design Rule Fixing is skipped.

The **set_local_link_library** command sets the **local_link_library** attribute on a design. The **local_link_library** attribute contains a list of design and library files that are searched before the files are specified with the **link_library** variable whenever a link operation is performed. The **target_library** variable specifies a technology library or a list of technology libraries containing the components (usually from a specific ASIC supplier) to use during optimization. The **compile** command sets the **local_link_library** attribute of the top-level design to the value of the **target_library** variable.

If **compile** is interrupted by an interrupt signal during an interactive process, it displays the following menu:

```
Please type in one of the following options:
  1 to Write out the current state of the design
  2 to Abort optimization
  3 to Kill the process
  4 to Continue optimization
Please enter a number:
```

If you select option 1, the design is written out and the menu reappears.

If the process is running in the background, you cannot use the menu. The number of interrupt signals determines what action is performed. The sequence is as follows:

```
^C Information: Preparing to interrupt optimization... (INT-5)
^C Information: Aborting Optimization... (INT-6)
^C Information: Process terminated by interrupt. (INT-4)
```

The interrupt signal can be different from one machine to another.

EXAMPLES

The following examples apply to all modes.

The following command specifies that the current design be optimized without mapping the logic:

```
prompt> compile -no_map
```

The following command optimizes the design TEST twice; once for speed and once for area:

```
prompt> current_design TEST

prompt> set_max_delay 2.0 -to [all_outputs]

prompt> compile

prompt> set_max_area 250

prompt> compile
```

The following example specifies that the command perform restricted optimizations across hierarchical boundaries:

```
prompt> compile -boundary_optimization
```

The following example demonstrates a simple optimization strategy. The commands in the example first define the current design. Design attributes and constraints are set. Optimization phases for **compile** are defined. The **compile** command is executed with the specified options.

```
prompt> current_design TRIAL
prompt> set_wire_load_model "10x10"
prompt> set_operating_conditions "WCCOM"
prompt> create_clock -period 12 \
    -waveform {0 6} CLK
prompt> set_input_delay 2 [all_inputs]
prompt> set_output_delay 3 [all_outputs]
prompt> set_max_area 580
prompt> set_structure -boolean false
prompt> set_flatten -phase true -effort medium
prompt> compile -incremental_mapping \
    -no_design_rule
prompt> report_constraint
```

SEE ALSO

```
alib_analyze_libs(2)
check_design(2)
create_clock(2)
insert_dft(2)
report_compile_options(2)
reset_design(2)
set_boundary_optimization(2)
set_cost_priority(2)
set_critical_range(2)
set_dont_touch(2)
set_dont_touch_network(2)
set_dont_use(2)
set_drive(2)
set_equal(2)
set_fix_hold(2)
set_fix_multiple_port_nets(2)
set_flatten(2)
set_fsm_encoding(2)
set_fsm_encoding_style(2)
set_fsm_order(2)
```

```
set_fsm_state_vector(2)
set_input_delay(2)
set_load(2)
set_local_link_library(2)
set_logic_dc(2)
set_logic_one(2)
set_logic_zero(2)
set_max_area(2)
set_max_capacitance(2)
set_max_delay(2)
set_max_fanout(2)
set_max_transition(2)
set_min_delay(2)
set_operating_conditions(2)
set_opposite(2)
set_output_delay(2)
set_prefer(2)
set_register_type(2)
set_structure(2)
set_timing_ranges(2)
set_unconnected(2)
set_wire_load_mode(2)
set_wire_load_model(2)
set_wire_load_min_block_size(2)
set_wire_load_selection_group(2)
ungroup(2)
uniquify(2)
alib_library_analysis_path(3)
compile_auto_ungroup_area_num_cells(3)
compile_auto_ungroup_count_leaf_cells(3)
compile_auto_ungroup_override_wlm(3)
compile_log_format(3)
compile_top_all_paths(3)
ungroup_keep_original_design(3)
```

compile_exploration

Performs a fast mapping of gates with limited timing and area optimization.

SYNTAX

```
status compile_exploration
[-check_only]
[-exact_map]
[-gate_clock]
[-no_autoungroup]
[-no_boundary_optimization]
[-no_seq_output_inversion]
[-scan]
```

ARGUMENTS

-check_only

Checks whether the design and libraries have all the data that the **compile_exploration** command requires to run.

-exact_map

Specifies that sequential cells are mapped exactly as indicated in the HDL code. Specifying this option does not mean that QN pins are not to be used in the mapped sequential element. Specifying the **-exact_map** option disables sequential output inversion. For more information, see the **compile_seqmap_enable_output_inversion** variable.

-gate_clock

Enables clock-gating optimization, which automatically removes or inserts clock gates. When this option is used with the **-exact_map** option, the command might not honor the **-exact_map** option for those registers that are involved with clock-gating optimization.

A clock-gating cell is not modified or removed if the cell or its parent hierarchical cell is marked with the **dont_touch** attribute by the the **set_dont_touch** command.

-no_autoungroup

Specifies to disable automatic ungrouping completely. All hierarchies are preserved unless otherwise specified.

-no_boundary_optimization

Specifies not to perform any hierarchical boundary optimization. By default, boundary optimization is turned on during the **compile_exploration** step.

-no_seq_output_inversion

Disables sequential output inversion. The sequential phase of all sequential elements is the same as in the RTL. When this

option is not specified, the **compile_exploration** command can invert sequential elements during mapping and optimization.

-scan

Enables the examination of the impact of scan insertion on mission-mode constraints during optimization as in a normal compile. Use this option to replace all sequential elements during optimization. Some scan-replaced sequential cells might be converted to nonscan cells later in the test synthesis process because of test design rule violations or explicit specifications.

DESCRIPTION

The **compile_exploration** command performs a fast mapping of gates with limited timing and area optimization. It typically runs much faster than the **compile** or **compile_ultra** command. Like the **compile** command, the optimization is controlled by constraints that you specify for the design. The **compile_exploration** command is targeted for high-performance designs with very tight timing constraints. It provides you with a simple approach to achieve critical delay optimization.

When used with the **set_host_options** command, the **compile_exploration** command uses up to the specified number of CPU cores on the same computer for parallel execution. For more details, see the description of the **-max_cores** option of the **set_host_options** man page.

This command can be used in the same manner as the **compile** command.

By default, the **compile_exploration** command incorporates two ungrouping phases for design hierarchies. The first phase occurs before the mapping step of first pass and attempts to ungroup small design hierarchies. The second ungrouping phase occurs during the mapping optimization step and applies a delay-based ungrouping strategy for design hierarchies. If you need to preserve all design hierarchies, specify the **-no_autoungroup** option.

By default, if `dw_foundation.sldb` is not specified in the list of the **synthetic_library** variable and the DesignWare license is checked out successfully, `dw_foundation.sldb` is automatically added to the synthetic library to improve the QoR by using the licensed DesignWare architectures. This behavior occurs in the current command only, and it does not affect the user-specified synthetic library and link library lists.

By default, the **compile_exploration** command performs hierarchical boundary optimization, which can change the function of the design. Therefore, hierarchical boundary optimization can only operate in its current environment. If input or output ports are complemented as a result of this optimization, port names are changed according to the setting of the **port_complement_naming_style** variable. To disable hierarchical boundary optimization, specify the **-no_boundary_optimization** option.

SEE ALSO

`compile(2)`
`compile_ultra(2)`

compile_partitions

Distributes compile jobs for a design.

SYNTAX

```
status compile_partitions
      -destination pass
```

Data Types

pass string

ARGUMENTS

-destination *pass*

Specifies which Automated Chip Synthesis pass to use for compile jobs. The command uses the *pass* value to generate the subdirectory name.

DESCRIPTION

This command distributes the compile jobs for a design by using the specified Automated Chip Synthesis destination pass directory. The command compiles the design by using the makefile located in *pass**n*/Makefile. The command uses the make utility tool specified by the **acs_make_exec** variable with optional arguments specified by using the **acs_make_args** variable. The **acs_num_parallel_jobs** variable specifies the number of compile jobs to run in parallel. The command saves a log of the run in the *pass**n*/logs/*\$acs_make_exec*.log file.

When using the Load Sharing Facility (LSF) software, you must define the root of the LSF installation location by using the **LSF_TOP** UNIX variable. This command requires that the LSF executable be located in *\$LSF_TOP/\$sh_arch/bin/lsmake*.

SEE ALSO

[write_makefile\(2\)](#)
[sh_arch\(3\)](#)

compile_prefer_runtime

Overrides runtime-intensive user settings with settings designed to improve runtime. The **compile_prefer_runtime** command settings take effect when you run the **compile_ultra**, **compile_ultra -incremental**, or the **optimize_netlist -area** command.

SYNTAX

status **compile_prefer_runtime**

ARGUMENTS

This command has no arguments.

DESCRIPTION

The **compile_prefer_runtime** command overrides runtime-intensive user settings with settings designed to improve runtime. The user-specified settings are reverted back to their original settings at the end of the optimization flow. The command also sets some internal variables to reduce runtime. The command overrides the user settings before optimization starts during **compile_ultra** and **optimize_netlist** and before **report_congestion**. The command is supported only in topographical mode.

Use the **compile_prefer_runtime** command only if runtime is a concern. The command disables runtime-intense optimizations, and therefore it might impact QOR. The effect of the setting changes depends on the design. You might not see runtime improvements on all designs.

For detailed information about the values that are changed when you use the **compile_prefer_runtime** command, see the "Using Topographical Technology" chapter in the *Design Compiler User Guide*.

EXAMPLES

The following example enables runtime mode settings during compile:

```
prompt> compile_prefer_runtime  
prompt> compile_ultra
```

SEE ALSO

`compile_ultra(2)`
`optimize_netlist(2)`
`report_congestion(2)`

compile_ultra

Performs a high-effort compile on the current design for better quality of results (QoR).

SYNTAX

```
status compile_ultra  
  [-incremental]  
  [-scan]  
  [-exact_map]  
  [-no_autoungroup]  
  [-no_seq_output_inversion]  
  [-no_boundary_optimization]  
  [-no_design_rule | -only_design_rule]  
  [-timing_high_effort_script  
   | -area_high_effort_script]  
  [-top]  
  [-retime]  
  [-gate_clock]  
  [-self_gating]  
  [-check_only]  
  [-congestion]  
  [-spg]  
  [-no_auto_layer_optimization]
```

ARGUMENTS

-incremental

Runs **compile_ultra** in incremental mode. In the incremental mode, the tool does not run the mapping or implementation selection stages.

-scan

Enables the examination of the impact of scan insertion on mission-mode constraints during optimization, as in a normal compile. Use this option to replace all sequential elements during optimization. Some scan-replaced sequential cells can be converted to nonscan cells later in the test synthesis process because of test design rule violations or explicit specifications.

-exact_map

Specifies that sequential cells are mapped exactly as indicated in the HDL code. Use of the **-exact_map** option does not mean the QN pin won't be used in the mapped sequential element. Use of the **-exact_map** option would disable sequential output inversion. For more information, see **compile_seqmap_enable_output_inversion**.

-no_autoungroup

Specifies that automatic ungrouping is completely disabled. All hierarchies are preserved unless otherwise specified.

-no_seq_output_inversion

Disables sequential output inversion. The phase sequential of all sequential elements is the same as in the RTL. Without this option, **compile_ultra** is free to invert sequential elements during mapping and optimization. For more information, see the man page for the **compile_seqmap_enable_output_inversion** variable.

-no_boundary_optimization

Specifies that no hierarchical boundary optimization is to be performed. By default, boundary optimization is turned on during **compile_ultra** activity.

Specifies that no hierarchical boundary optimization is to be performed. By default, the **-no_boundary_optimization** option does not disable constant propagation across design hierarchies. To disable constant propagation with the **-no_boundary_optimization** option, set the **compile_enable_constant_propagation_with_no_boundary_opt** variable to **false**.

This option sets the **compile_preserve_subdesign_interfaces** variable to **true**.

-no_design_rule

Determines whether the command fixes design rule violations before exiting. The **-no_design_rule** option specifies for the command to exit before fixing design rule violations, thus allowing you to check the results in a constraint report before fixing the violations. The default is to perform both design rule fixing and mapping optimizations before exiting.

The **-no_design_rule** and **-only_design_rule** options are mutually exclusive. Use only one option.

-only_design_rule

Determines whether the command fixes design rule violations before exiting. The **-only_design_rule** option specifies for the command to perform only design rule fixing; that is, mapping optimizations are not performed. The default is to perform both design rule fixing and mapping optimizations before exiting.

The **-no_design_rule** and **-only_design_rule** options are mutually exclusive. Use only one option. The **-only_design_rule** option can be used only with the **-incremental** option.

-timing_high_effort_script

This option is available in the tool to support backward compatibility with existing scripts and is ignored for optimization purposes.

-area_high_effort_script

This option is available in the tool to support backward compatibility with existing scripts and is ignored for optimization purposes.

-top

Fixes design rule and top-level timing violations in a design. By default, this option fixes all design rule violations, but only those timing violations whose paths cross top-level hierarchical boundaries. If you want this option to fix timing violations for all paths, set the **compile_top_all_paths** variable to **true**.

-retime

Uses the adaptive retiming algorithm during optimization to improve delay. This option is ignored if the **-only_design_rule** option or the **-top** option is chosen at the same time.

-gate_clock

Enables clock gating optimization: clock gates are automatically inserted or removed. The **-gate_clock** option cannot be used in combination with the **-only_design_rule** option. When used with the **-exact_map** option, it might not be possible to honor the **-exact_map** option for those registers that are involved with clock gating optimization.

A clock gating cell is not modified or removed if it or its parent hierarchical cell is marked **dont_touch** with the **set_dont_touch**

command.

-self_gating

Enables the execution of XOR self-gating insertion.

Self-gating is an XOR based clock gating technique that optimizes dynamic power by gating the clock signal in those cycles in which the data saved in a register remains unchanged. An enable condition is computed by comparing the stored data with the new data arriving at the data pin, and that signal is used to drive the inserted self-gating cell.

A self-gating cell can be shared across several registers by creating a combined enable condition so that the area and power overhead due to the inserted cells is minimized.

The selection of registers to be gated and the grouping of them to form the self-gating banks are driven by the switching activity at the registers' data pins, the timing slack available, and the physical proximity between the registers to be grouped.

This option is only supported in Design Compiler topographical mode.

The **-self_gating** option cannot be used in combination with the **-only_design_rule** option.

-check_only

Checks whether the design and libraries have all the data that **compile_ultra** requires to run successfully. This option is available only in Design Compiler topographical mode.

-congestion

This option will be obsolete in a future release. See the **-spg** option for information about congestion optimization.

-spg

This option is available only in Design Compiler topographical mode. Enables physical guidance, congestion optimization, and automatic layer optimization. Congestion optimization reduces routing-related congestion. Physical guidance enables Design Compiler Graphical to save coarse placement information and pass this coarse placement information to IC Compiler. With this coarse placement, IC Compiler can begin the implementation flow with the **place_opt** command.

IC Compiler no longer needs to re-create the coarse placement by running commands such as **create_placement**, **remove_buffer_tree**, or **psynopt**. By using the Design Compiler coarse placement as a starting point for placement, runtime and area correlation with IC Compiler are improved.

Design Compiler Graphical automatically performs layer-aware optimization when you use the **-spg** option, modeling parasitic variation across metal layers in a way that benefits optimization. This optimization helps remove excess pessimism, leading to better area and power.

In addition to the default layer-aware optimization, you can also specify net constraints for layer optimization by setting specific constraints using the **set_net_routing_layer_constraints** command or by creating a net-search pattern.

In the net-search pattern approach, you define a net-search pattern by using the **create_net_search_pattern** command and then define associated minimum and maximum routing layer constraints for the search pattern by using the **set_net_search_pattern_delay_estimation_options** command. Design Compiler invokes net-pattern identification after the high-fanout synthesis step in **compile_ultra** and assigns the minimum and maximum constraints to the matching nets. The subsequent optimizations consider the effects of the constraints (for example, the unit resistance and capacitance values of matching nets will change) during buffering and buffer removal. You can define as many net-search patterns and associated layer constraints as needed. In general, however, it is recommended to start with very long nets (for example, 500 um) with top routing layers (for example, M7 and M8). You should consider this approach when your design shows significant unit resistance variation (see RCEX-011 resistance values) across all available routing layers.

Note that the user-constraints and net-pattern layer optimization methods might affect runtime.

-no_auto_layer_optimization

The **-no_auto_layer_optimization** option is obsolete. Running the option has no effect on layer optimization.

DESCRIPTION

The **compile_ultra** command performs a high-effort compile on the current design for better quality of results (QoR). As with the **compile** command, optimization is controlled by constraints that you specify on the design. This command is targeted toward high-performance designs with very tight timing constraints. It provides you with a simple approach to achieve critical delay optimization. The **compile_ultra** command packages all the DC Ultra features and enables them by default. It requires a DC Ultra license plus a DesignWare Foundation license. This command provides the best strategy for optimum overall QoR and performance.

When used in conjunction with the **set_host_options** command, **compile_ultra** uses up to the user-specified number of CPU cores on the same computer for parallel execution. See the description of the **-max_cores** option in the **set_host_options** man page for more information.

This command can be used in the same manner as the **compile** command.

By default, **compile_ultra** incorporates two ungrouping phases for design hierarchies. The first phase is performed before "Pass1 Mapping" and attempts to ungroup small design hierarchies. This first ungrouping phase can be turned off using the following command:

```
set compile_ultra_ungroup_small_hierarchies false
```

The second ungrouping phase is performed during "Mapping Optimization" and applies a delay-based ungrouping strategy for design hierarchies. If you need to preserve all design hierarchies, use the **-no_autoungroup** option. If you want to preserve the hierarchies for a specific block, use the **set_ungroup** command.

By default, if `dw_foundation.sldb` is not in the `synthetic_library` list, and the DesignWare license is successfully checked out, `dw_foundation.sldb` is automatically added to the `synthetic_library` to use the QoR benefit provided by the licensed DesignWare architectures. This behavior occurs in the current command only, and it does not affect the user-specified **synthetic_library** and **link_library** list.

By default, all DesignWare hierarchies are unconditionally ungrouped in the second pass of compile. You can set the **compile_ultra_ungroup_dw** variable to control the ungrouping process of DesignWare components.

By default, hierarchical boundary optimization is performed on the design. This can change the function of the design so that it can operate only in its current environment. If input or output ports are complemented as a result of this optimization, port names are changed according to the **port_complement_naming_style** variable. Use the **-no_boundary_optimization** option to turn off the boundary optimization feature.

By default, the tool applies a compile strategy intended to improve the resulting area of the design, possibly at the cost of additional runtime. The strategy can make changes to variables or constraints that modify **compile_ultra** behavior and perform additional passes to achieve better area.

EXAMPLES

The following example turns off boundary optimization for cell U1:

```
prompt> set_boundary_optimization [get_cells U1] false
prompt> compile_ultra
```

SEE ALSO

compile(2)
set_host_options(2)
create_net_search_pattern(2)
set_net_search_pattern_delay_estimation_options(2)

compute_polygons

Returns a list or collection of polygons that exactly cover the region computed by performing a Boolean operation on the input polygons.

SYNTAX

```
list compute_polygons  
-boolean and | or | not | xor  
poly_list1  
poly_list2
```

Data Types

poly_list1 list or collection
poly_list2 list or collection

ARGUMENTS

-boolean and | or | not | xor

Specifies the Boolean operation to perform on the two input polygon lists or collections to produce the output polygon list or collection.

"**and**" finds the area occupied by both the first and second input polygon list or collection.

"**or**" finds the area occupied by either the first or second input polygon list or collection, or occupied by both.

"**not**" finds the area occupied by the first input polygon list or collection and not occupied by the second polygon list or collection.

"**xor**" finds the area occupied by either the first or second input polygon list or collection, but not both.

poly_list1

Specifies the first list or collection of input polygons. You can specify a single polygon or multiple polygons, representing the first argument for the geometric Boolean operation.

To specify a polygon as a list, enter a sequence of X-Y coordinate pairs that represent the successive vertices of the polygon, using the following format:

```
{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}
```

The coordinate unit size is specified in the technology file; typically it is microns.

Polygons must be rectilinear, so the X coordinate of each data point must match the X coordinate of a neighboring data point, and the Y coordinate must match the Y coordinate of the other neighboring data point. The last point in the list must be the same as the first.

Instead of specifying lists of coordinates, you can specify a polygon collection. In that case, the output of the command is also a collection.

poly_list2

Specifies the second list or collection of input polygons. The command performs a Boolean operation between the first and the second list or collection of polygons.

DESCRIPTION

This command performs a Boolean operation (OR, AND, NOT, or XOR) between two polygons or between two sets of polygons, specified as lists of vertices or as polygon collections. It returns the result in the same form as the input polygons, either a list or a polygon collection. The result can be a single polygon or a set of multiple, disjoint polygons.

Before using this command, the Milkyway design library containing the input polygons must be open. The input polygons must be rectilinear, consisting of only vertical and horizontal line segments that meet at right angles.

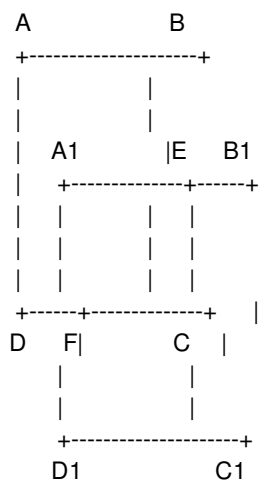
You can directly pass the output of this command as a parameter directly to another **compute_polygons** command, but not to the other polygon commands, **convert_from_polygon**, **resize_polygon**, or **get_polygon_area**. This is because the other polygon commands can only take a single polygon as input. Instead, you must use a Tcl list command such as the **foreach** or **lindex** command to extract each polygon from the returned list, and then pass each polygon to the other polygon command.

To view the coordinates of a polygon collection stored in a variable, use the following command:

```
prompt> get_attribute $variable_name coordinate
```

EXAMPLES

Assume that you have two polygons, A-B-C-D-A and A1-B1-C1-D1-A1, as shown in the following figure:



The following example performs the Boolean OR operation on these input polygons. The result is the B-E-B1-C1-D1-F-D-A-B polygon.

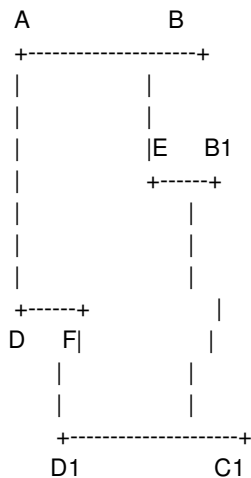
```
prompt> compute_polygons -boolean or \
```

```

{{0 30} {30 30} {30 10} {0 10} {0 30}} \
{{10 20} {40 20} {40 0} {10 0} {10 20}}
{{30.000 30.000} {30.000 20.000} {40.000 20.000} {40.000 0.000} {10.000 0.000}
{10.000 10.000} {0.000 10.000} {0.000 30.000} {30.000 30.000}}

```

The following figure shows the resulting polygon.



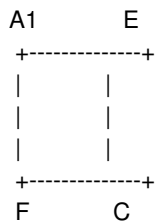
The following example performs the Boolean AND operation on the same input polygons. The result is the A1-E-C-F-A1 polygon.

```

prompt> compute_polygons -boolean and \
{{0 30} {30 30} {30 10} {0 10} {0 30}} \
{{10 20} {40 20} {40 0} {10 0} {10 20}}
{{10.000 20.000} {30.000 20.000} {30.000 10.000}
{10.000 10.000} {10.000 20.000}}

```

The following figure shows the resulting polygon.



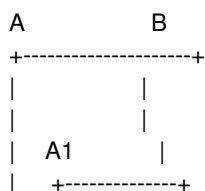
The following example performs the Boolean NOT operation on these input polygons. The result is the B-E-A1-F-D-A-B polygon, which is the region covered by the first polygon, A-B-C-D-A, but not covered by the second polygon, A1-B1-C1-D1-A1.

```

prompt> compute_polygons -boolean not \
{{0 30} {30 30} {30 10} {0 10} {0 30}} \
{{10 20} {40 20} {40 0} {10 0} {10 20}}
{{30.000 30.000} {30.000 20.000} {10.000 20.000} {10.000 10.000}
{0.000 10.000} {0.000 30.000} {30.000 30.000}}

```

The following figure shows the resulting polygon.




```

|   |   E
|   |
|   |
+-----+
D   F

```

The following example performs the Boolean XOR operation on these input polygons. The result is two polygons, B-E-A1-F-D-A-B and B1-C1-D1-F-C-E-B1, which are the areas covered by either input polygon, but not both.

```

prompt> compute_polygons -boolean xor \
{{0 30} {30 30} {30 10} {0 10} {0 30}} \
{{10 20} {40 20} {40 0} {10 0} {10 20}}
{{30.000 30.000} {30.000 20.000} {10.000 20.000} {10.000 10.000} {0.000 10.000}
{0.000 30.000} {30.000 30.000}} {{40.000 20.000} {40.000 0.000} {10.000 0.000}
{10.000 10.000} {30.000 10.000} {30.000 20.000} {40.000 20.000}}

```

The following figure shows the resulting polygons, moved apart slightly to clearly show the disjoint regions.

```

A           B
+-----+
|         |
|         |
|  A1     | E  B1
|  +-----+ +-----+
|  |       | E |  |
|  |       | |  |
|  | F     | |  |
+----+ +-----+ +-----+
D  F |       C  |
    |         |
    |         |
    +-----+
    D1        C1

```

The following example shows how **compute_polygons** can accept list of polygons as input.

```

prompt> compute_polygons -boolean or {{{0 0} {10 0} {10 10} \
{0 10} {0 0}} {{20 0} {30 0} {30 10} {20 10} {20 0}}} \
{{{10 0} {20 0} {20 10} {10 10} {10 0}}
{{0.000 10.000} {30.000 10.000} {30.000 0.000} {0.000 0.000} {0.000 10.000}}

```

SEE ALSO

convert_from_polygon(2)
resize_polygon(2)

connect_logic_net

Connects a logic net to logic ports in a UPF description.

SYNTAX

```
status connect_logic_net  
    net_name  
    -ports port_list  
    [-reconnect]
```

Data Types

```
net_name    string  
port_list   list
```

ARGUMENTS

net_name

Specifies the net name. You must use a simple name.

-ports *port_list*

Specifies the ports connected to the net. The ports must be on the interface of the active UPF scope or on the design elements that are located in the active scope and its descendants.

-reconnect *reconnect*

Allows a port that is already connected to a net to be disconnected from the existing net and connected to the newly specified net.

DESCRIPTION

This command connects a logic net to the specified ports. The net is propagated through implicitly created ports and nets throughout the logic hierarchy in the descendant tree of the active UPF scope. The connection from the specified net in the active UPF scope to any element in the port list cannot cross any power-domain boundaries. The -reconnect option disconnects any existing net from the specified port and connects the newly specified net to all the ports present in the port_list.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

`create_logic_port(2)`

`create_logic_net(2)`

connect_net

Connects the specified net to the specified pins or ports.

SYNTAX

```
status connect_net  
    net  
    object_list
```

Data Types

```
net          string  
object_list list
```

ARGUMENTS

net

Specifies the net to connect. The net must be a scalar (single bit) net, and must exist in the current design.

object_list

Specifies a list of pins and ports to which the net is to be connected. Pins and ports must be at the same hierarchical level as the specified net, and must exist in the current design. If a specified pin or port is already connected, the tool issues an error message.

DESCRIPTION

This command connects a net to the specified pins or ports at the same hierarchical level. The net can be at any level of hierarchy but the pins or ports must be at the same level. A net can be connected to many pins or ports; however, you cannot connect a pin or port to more than one net.

To disconnect objects on a net, use the **disconnect_net** command.

To display pins and ports on a net, use either the **all_connected** or **get_nets -of \$net** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **connect_net** to connect net NET0 to ports A1 and A2 and pin U1/A. The **all_connected** command returns the objects connected to net NET0.

```
prompt> connect_net NET0 [get_ports {A1 A2}]
```

```
prompt> connect_net NET0 [get_pins U1/A]
```

```
prompt> all_connected NET0  
{A1 A2 U1/A}
```

The following example shows the error message generated when you attempt to connect a pin or port to more than one net:

```
prompt> connect_net MY_NET_1 PORT1  
Connecting net 'MY_NET_1' to port 'PORT1'.
```

```
prompt> connect_net MY_NET_2 PORT1  
Error: Object 'PORT1' is already connected to net 'MY_NET_1'.
```

The following example shows how **connect_net** is used to connect the U1/MY_NET net to the U1/MY_PORT port:

```
prompt> connect_net U1/MY_NET U1/MY_PORT
```

SEE ALSO

- all_connected(2)
- create_net(2)
- current_design(2)
- disconnect_net(2)
- remove_net(2)

connect_pin

Connects pins or ports at any level of hierarchy.

SYNTAX

```
status connect_pin  
-from from_object  
-to to_list  
[-port_name port_name]  
[-verbose]
```

Data Types

```
from_object  collection  
to_list      collection  
port_name    string
```

ARGUMENTS

-from *from_object*

Specifies the pin or port from which to make the connection. The direction of the pin or port cannot be inout.

-to *to_list*

Specifies the pins and ports to which to make the connection. The pins and ports can be at any level of the hierarchy. The direction of the pins or ports cannot be inout.

-port_name *port_name*

Specifies the base name to use as the names of ports that are created on subdesigns to make the connection.

-verbose

Specifies that the tool displays individual netlist operations while making the global connections.

DESCRIPTION

This command performs global connections between the source object specified in the **-from** option and the objects specified in the **-to** option. The specified pins and ports can be at any level of hierarchy. The parent design of the pins and ports must be unique.

While making the global connections, ports and nets are created in the subdesigns, if needed. Ports in the subdesigns are reused

regardless of their names, when the from pin is already connected to a net. Also, a port in a subdesign is reused if it is unconnected and the name is as specified by the **-port_name** option.

Wherever applicable, the name of the net that is created is the name of the connecting port, as long as there is no net with the same name in that design. If there is an existing net with the name, the name of the net is generated.

The **connect_pin** command ensures that multiply-driven nets are not created. The command does not allow a connection from an output pin to an output pin.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **connect_pin** command to connect the U1/Z output pin to the mid1/bot1/U3/A input pin.

Use the **report_net -connections** or **report_cell -connections** command to view the connections.

```
prompt> connect_pin -from [get_pins U1/Z] \  
-to [get_pins mid1/bot1/U3/A]
```

```
prompt> report_net -connections [get_net -of [get_pins U1/Z]]
```

SEE ALSO

- all_connected(2)
- connect_net(2)
- create_port(2)
- report_cell(2)
- report_net(2)

connect_supply_net

Connects the supply net to specified supply ports and pins.

SYNTAX

```
status connect_supply_net  
    supply_net_name  
    -ports list  
    [-vct vct_name]
```

Data Types

```
supply_net_name    string  
list                list  
vct_name           string
```

ARGUMENTS

supply_net_name

Specifies the name of the supply net to be connected, which must be an existing simple (nonhierarchical) supply net name.

-ports *list*

Specifies the supply ports or pins that are to be connected with the supply net. Each item in the list is the hierarchical name of a supply port or pin.

-vct *vct_name*

Specifies the value conversion table (VCT) to be used in this connection.

The tool supports only the following 14 predefined VCT names: UPF2VHDL_SL ; UPF_GNDZERO2VHDL_SL ; UPF2SV_LOGIC ; UPF_GNDZERO2SV_LOGIC ; VHDL_TIED_HI ; SV_TIED_HI ; VHDL_TIED_LO ; SV_TIED_LO ; VHDL_SL2UPF ; VHDL_SL2UPF_GNDZERO ; SV_LOGIC2UPF ; SV_LOGIC2UPF_GNDZERO ; SV_LOGIC2UPF_MD ; SV_LOGIC2UPF_GNDZERO_MD.

DESCRIPTION

The **connect_supply_net** command makes an explicit connection of a supply net to specified supply ports or pins. It overrides (has higher precedence than) the automatic connection semantics that might otherwise apply.

If a design element is not connected explicitly to any supply net using the **connect_supply_net** command, it shares the primary

power or ground supply net with the power domain to which it belongs.

Before they can be connected, the supply net must be created in the same power domain as the supply port. The instance that contains the pin must also be in the extent of the same power domain.

The command can be used to connect a supply net to bias pins. Bias pins that meet following conditions are supported:

- The bias pin is from a macro cell
- The bias pin direction is "input" or "inout"
- The bias pin physical_connection is "routing_pin"

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example connects a supply net named VSS with the supply port VSS:

```
prompt> create_power_domain PD1 -elements INST_1
PD1
prompt> create_supply_net VSS -domain PD1
VSS
prompt> create_supply_port VSS -domain PD1
VSS
prompt> connect_supply_net VSS -ports VSS
1
```

The following example shows how to connect a supply set with a supply port:

```
prompt> connect_supply_net sset.power -ports VDD
1
prompt> connect_supply_net sset.ground -ports VSS
1
```

SEE ALSO

create_supply_net(2)
report_supply_net(2)

continue

Begins the next loop iteration.

SYNTAX

int **continue**

ARGUMENTS

None

DESCRIPTION

This command begins the next iteration of the innermost loop. To immediately reevaluate the condition of a while loop, use the **continue** command, rather than executing the remaining statements to the end.

The **continue** command always returns the integer 1, indicating successful operation. A syntax error is reported if the **continue** command is used outside a loop structure.

EXAMPLES

The following example plots the first 10 sheets of the current design, except for sheet 5:

```
set p 0
while {$p <= 10} {
  if {$p % 2} {
    incr p
    continue
  }
  echo "$p squared is: [expr $p * $p]"; incr p
}
```

SEE ALSO

```
break(2)  
while(2)
```

convert_from_polygon

Converts each polygon into a list or collection of mutually exclusive rectangles.

SYNTAX

```
list convert_from_polygon
    polygons
    [-format polygon | rectangle]
```

Data Types

polygons polygon list or collection

ARGUMENTS

polygons

Specifies the list or collection of input polygons to be decomposed into rectangles. You can specify a single polygon or multiple polygons.

To specify a polygon as a list, enter a sequence of X-Y coordinate pairs that represent the successive vertices of the polygon, using the following format:

```
{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}
```

The coordinate unit size is specified in the technology file; typically it is microns.

Polygons must be rectilinear, so the X coordinate of each data point must match the X coordinate of a neighboring data point, and the Y coordinate must match the Y coordinate of the other neighboring data point. The last point in the list must be the same as the first.

Instead of specifying lists of coordinates, you can specify a polygon collection. In that case, the output of the command is also a collection.

-format polygon | rectangle

Specifies the output format of the resulting rectangles.

By default, each resulting rectangle is represented as two data points at opposite corners of the rectangle:

```
{x1 y1} {x2 y2}
```

If you specify **-format polygon**, each resulting rectangle is represented in polygon format, using five data points for the round trip around the perimeter of the rectangle:

```
{{x1 y1} {x2 y1} {x2 y2} {x1 y2} {x1 y1}}
```

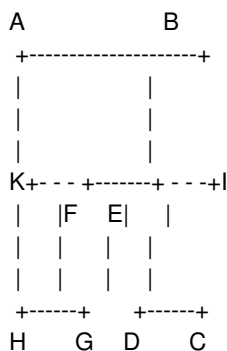
DESCRIPTION

This command converts a polygon into a list of mutually exclusive rectangles. The returned rectangles are represented as a list of coordinates or a collection. Before using this command, the Milkyway design library must be open.

Note that this command might return a list of disjoint polygons. Because the polygon commands take a single polygon as input, you cannot directly pass the result as a parameter to another polygon command. Instead, you must use a Tcl list command, such as the **foreach** or **lindex** command, to extract each polygon from the returned list, and then pass each polygon to the polygon command.

EXAMPLES

The following example converts the A-B-C-D-E-F-G-H-A polygon to three rectangles, K-F-G-H, E-I-C-D, and A-B-I-K.



```
prompt> convert_from_polygon {{0 20} {30 20} {30 0} \
{20 0} {20 10} {10 10} {10 0} {0 0} {0 20}}
{{0.000 0.000} {10.000 10.000}} {{20.000 0.000}
{30.000 10.000}} {{0.000 10.000} {30.000 20.000}}
```

SEE ALSO

convert_to_polygon(2)
compute_polygons(2)
resize_polygon(2)

convert_pg

Converts RTL PG information extracted from RTL into UPF. This command is supported only in UPF mode.

SYNTAX

```
status convert_pg
      [-net_creation_style style]
```

Data Types

style string

ARGUMENTS

-net_creation_style *style*

Optional argument which tells **convert_pg** the style of supply net to be created in case a matching supply net is not found in the UPF. Valid values for *style* are *domain_dependent* and *domain_independent*.

DESCRIPTION

The **convert_pg** command converts the PG network information extracted from RTL during the **link** command into UPF constraints. For the **link** command to extract the PG network from RTL, the **dc_allow_rtl_pg** Tcl Boolean variable must be set before reading in the RTL.

The PG network is converted into a set of domain-dependent UPF constraints or domain-independent UPF constraints depending on the *style* specified to the **-net_creation_style** option.

The **convert_pg** command requires some basic UPF constraints to have been loaded and at least a top level power domain defined already. It then tries to match each RTL PG net with an existing supply net, if any. If there is no match, then either a domain-dependent supply net or a domain-independent supply net is created for the RTL PG net, based on the value specified to the **-net_creation_style** option. In addition, supply ports are also created as required. If a supply net cannot be created because of the lack of knowledge of a target power domain, then errors are reported.

The RTL net connections to PG pins are converted to **/fBconnect_supply_net/fP** commands. If your UPF connects a different supply net to the same PG pins, then an error is reported.

If **convert_pg** command can successfully convert all of the PG network to UPF, then it returns a status of 1. Otherwise, a status of 0 is returned.

After the **convert_pg** command converts the PG network to UPF successfully, subsequent calls to the **convert_pg** command yield nothing.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples show different usages of the **convert_pg** command and its outcome.

The following shows the behavior when the **convert_pg** command is called without any RTL PG information.

```
prompt> convert_pg  
Information: The design does not contain any RTL-derived PG nets. (UPF-439)
```

The following shows the behavior when the **convert_pg** command is called without loading any UPF constraints.

```
prompt> convert_pg  
Error: No top level power domain found.
```

The following shows the behavior when the **convert_pg** command is called again after a successful call.

```
prompt> convert_pg  
Information: The design does not contain any RTL-derived PG nets. (UPF-439)
```

The following shows the behavior when the **convert_pg** command successfully converts the PG network into domain-dependent UPF.

```
prompt> convert_pg -net_creation_style domain_dependent  
Generating UPF for PG netlist derived from RTL:  
  
create_supply_net {vdd} -domain {TOP}  
create_supply_port {vdd} -direction {in}  
connect_supply_net {vdd} -port {vdd}  
connect_supply_net {vdd} -port {mid1/U1/VDD mid1/bot1/U1/VDD}  
1
```

SEE ALSO

dc_allow_rtl_pg(3)

convert_to_polygon

Returns a polygon list for the specified objects.

SYNTAX

```
list convert_to_polygon  
  [-quiet]  
  [-collection]  
  object_spec
```

Data Types

object_spec object collection

ARGUMENTS

-quiet

Suppresses the reporting of error and warning messages.

-collection

This option is not supported in Design Compiler.

object_spec

Specifies the input object collection. The object class must be fixed cells, fixed physical-only cells, a placement blockage, or a move bound.

DESCRIPTION

This command returns polygons calculated from the input objects. The returned polygon is represented as a list of polygon vertex coordinates.

EXAMPLES

The following example returns a polygon list from the MB#123 move bound.


```
prompt> convert_to_polygon [get_bounds MB#123]  
{{99.490 111.340} {117.590 111.340} {117.590 112.620} {99.490 112.62}  
{99.490 111.340}}
```

SEE ALSO

convert_from_polygon(2)
compute_polygons(2)
resize_polygon(2)

copy_collection

Duplicates the contents of a collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

```
collection copy_collection  
collection1
```

Data Types

```
collection1  collection
```

ARGUMENTS

collection1

Specifies the collection to be copied. If an empty string is used for the *collection1* argument, the command returns the empty string (a copy of the empty collection is an empty collection).

DESCRIPTION

This command is no longer needed and is provided only to make old scripts work without modification.

EXAMPLES

The following example from PrimeTime shows the result of copying a collection. Functionally, it is not much different that having multiple references to the same collection, except it is slower.

```
prompt> set c1 [get_cells "U1*"]  
{U1 U10 U11 U12}  
prompt> set c2 [copy_collection $c1]  
{U1 U10 U11 U12}  
prompt> unset c1  
prompt> query_objects $c2  
{U1 U10 U11 U12}
```

SEE ALSO

`collections(2)`

copy_design

Copies a design to a new design, or copies a list of designs to a new file in dc_shell memory.

SYNTAX

```
string copy_design  
    design_list  
    target_name
```

Data Types

```
design_list  string  
target_name string
```

ARGUMENTS

design_list

Specifies a lists of designs to copy. If a design is specified, the *target_name* must be a file name. All of the listed designs are copied to the the specified file with their base design names.

target_name

Specifies the name of the new design to create or the name of the file to which multiple designs are copied.

DESCRIPTION

The **copy_design** command copies a list of designs either to a target file or to a newly created design.

In the first form, the **copy_design** command copies the contents of *design_name* to *target_name*. The attributes and constraints of *design_name* are copied with the design. A design cannot be copied over another design that already exists in dc_shell.

In the second form, **copy_design** copies the designs listed in *design_name* to the file specified by *target_name*.

To write these files to disk, use the **write** command.

EXAMPLES

This example copies the design named *TEST* to the design named *BILL*:

```
prompt> copy_design TEST BILL
```

```
Information: Copying design /usr/example/TEST.DB:TEST to /usr/example/TEST.db:BILL.  
(UIMG-45)
```

```
prompt> list_designs
```

```
BILL TEST.db (*)
```

This example copies design *TEST* to the new file *my_test.db*:

```
prompt> copy_design TEST /usr/example/my_test.db:TEST
```

```
Information: Copying design /usr/example/TEST.db:TEST to /usr/example/my_test.db:TEST.  
(UIMG-45)
```

```
prompt> list_designs -show_file
```

```
/usr/example/TEST.db  
TEST (*)
```

```
/usr/example/my_test.db  
TEST
```

In this example, the **get_designs** command is used with **copy_design** to copy all of the designs currently read into *dc_shell* to a single file:

```
prompt> list_designs -show_file
```

```
/usr/example/TEST.db  
TEST (*)
```

```
/usr/example/ADDER.db  
ADDER
```

```
prompt> copy_design [get_designs *] /usr/dc/project/all.db
```

```
Information: Copying design /usr/example/TEST.db:TEST to /usr/dc/project/all.db:TEST.  
(UIMG-45)
```

```
Information: Copying design ADDER.db:ADDER to /usr/dc/project/all.db:ADDER.  
(UIMG-45)
```

```
prompt> list_designs -show_file
```

```
/usr/example/TEST.db  
TEST (*)
```

```
/usr/example/ADDER.db  
ADDER
```

```
/usr/dc/project/all.db  
ADDER TEST
```

If you attempt to copy multiple designs without specifying a target file name as the second argument of the **copy_design** command, an error is generated, as shown below:

```
prompt> copy_design {A, B, C} all.db:TEST3
```

```
Error: Cannot copy multiple designs to a single design  
(UIMG-4)
```

SEE ALSO

current_design(2)
get_designs(2)
remove_design(2)
rename_design(2)
write(2)

copy_mw_lib

Copies a Milkyway library to another location.

SYNTAX

```
status copy_mw_lib
[-from mw_lib]
-to lib_name
```

Data Types

```
mw_lib    string
lib_name  string
```

ARGUMENTS

-from *mw_lib*

Specifies the name of the source Milkyway library to be copied. The *mw_lib* value can be a library name or a collection of libraries. By default, the command uses the current Milkyway library.

-to *lib_name*

Specifies the destination Milkyway library name.

DESCRIPTION

The **copy_mw_lib** command copies a Milkyway library to another location. It returns a status indicating success or failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example copies a Milkyway library design to another location:

```
prompt> copy_mw_lib -from design -to design.bak
```

1

SEE ALSO

`rename_mw_lib(2)`

cputime

Reports the CPU time in seconds.

SYNTAX

```
status cputime  
  [-all]  
  [-verbose]
```

ARGUMENTS

-all

Reports the total CPU time of the main process and its child processes. This is the default; this option is kept for backward compatibility.

-verbose

Reports in detail the CPU time and elapsed (wall-clock) time of the main process and each child process, including placement, extraction, and routing.

DESCRIPTION

The **cputime** command reports the CPU time in seconds. If you specify **-verbose**, it reports the CPU time and elapsed (wall-clock) time for the main process and each child process. The runtime numbers are collected from the operating system.

When you use the **set_host_options** command, the tool runs multiple threads. Currently the operating system measures the CPU time by adding the usage for all threads. It does not take into account parallelization of the threads. This means that the report might show a larger CPU time than elapsed time when you use **set_host_options**. The elapsed time (wall-clock time) is more appropriate for measuring multicore runtime.

The elapsed time depends on many external factors and can vary for every session. It depends on factors such as the I/O traffic, the network traffic, RAM and swap usage, and the other processes running on the same machine. When the elapsed time is much longer than the CPU time, it might point out an inefficiency of the computing environment, such as insufficient memory, too many processes running on the same machine, or a slow network. The only way to make the elapsed time close to the CPU time is to run only one session on a machine with enough RAM and using the local disk.

EXAMPLES

The following example shows the default command output:

```
prompt> cputime  
1202
```

The following example shows the output when using the **-verbose** option:

```
prompt> cputime -verbose  
Main process CPU: 1202 s (0.33 hr) Elapse: 1800 s (0.50 hr)  
Child "placement" called 3 times, total CPU: 200 s (0.06 hr)  
  Elapse: 250 s (0.07 hr)  
Child "extraction" called 2 times, total CPU: 100 s (0.03 hr)  
  Elapse: 150 s (0.04 hr)  
Total CPU: 1502 s ( 0.42 hr) Elapse: 1800 s ( 0.50 hr)
```

```
1523
```

SEE ALSO

mem(2)
set_host_options(2)

create_auto_path_groups

Creates path groups for the current design.

SYNTAX

```
string create_auto_path_groups  
-mode rtl | mapped  
[-exclude IO | macro | ICG]  
[-slack slack]  
[-max max]  
[-min_regs_per_hierarchy min_regs]  
[-prefix prefix]  
[-file file_name]  
[-verbose]  
[-user_path_groups_file user_file_name]  
[-skip]
```

Data Types

<i>slack</i>	double
<i>max</i>	int
<i>min_regs</i>	int
<i>prefix</i>	string
<i>file_name</i>	string
<i>user_file_name</i>	string

ARGUMENTS

-mode rtl | mapped

Specifies if paths groups should be created for all hierarchies before optimization (**-mode rtl**) or only for hierarchies that fail timing after optimization (**-mode mapped**).

-exclude IO | macro | ICG

Prevents the generation of specific path groups. Supported values are **macro**, **IO**, and **ICG**. You can specify any combination of values. By default, the command creates path groups for I/Os, macros, and integrated clock-gating cells.

-slack *slack*

Specifies the slack value used to create group paths in mapped mode. The default is 0.0.

-max *max*

Specifies the maximum number of path groups in mapped mode. The default is 0, which means unlimited.

-min_regs_per_hierarchy min_regs

Specifies the minimum number of registers per hierarchy to be considered for path groups in **rtl** mode. The default is 10.

-prefix prefix

Specifies the prefix of the path groups names. The default is `synopsys_pg_`.

-file file_name

Writes **group_path** commands to the specified file.

-verbose

Activates the verbose mode.

-user_path_groups_file user_file_name

Specifies the file to which user path groups are saved. The default is `auto_path_groups.user_path_groups.tcl`.

DESCRIPTION

Creates path groups automatically for the current design. You can create path groups either after elaboration or after the initial compile.

To create path groups after the design has been elaborated, run the **create_auto_path_groups** command with the **-mode rtl** option. This creates one path group per hierarchy within the design. If there are multiple levels of hierarchy, the tool creates path groups for each level. The advantage of creating path groups at this stage is that the tool can perform all optimizations during the initial compile. However, the number of hierarchical levels within the design can be large, leading to a large number of path groups and causing excessive the runtime. Modules that have the **optimize_registers** attribute set on them are excluded to avoid the possibility of **optimize_registers** not working on the modules that have a path group set on them.

To create path groups after the design has successfully completed the initial compile, run the **create_auto_path_groups** command with the **-mode mapped** option. This creates one path group per hierarchy only for those hierarchies within the design that are not meeting timing. If there are multiple levels of hierarchy that are not meeting timing, the tool creates path groups for each of those levels. Fewer path groups are created when you use the **-mode mapped** option compared to the **-mode rtl** option because, in most cases, not all hierarchical blocks violate timing. Only those that fail to meet timing are assigned a separate path group, and therefore the runtime impact is reduced. In most cases, use this flow.

When you use the **-mode mapped** option, you can provide a slack value by specifying the **-slack** option or provide the maximum number of paths by specifying the **-max** option. The default slack value is zero, which means that the command creates path groups for all violating paths. The default maximum number of paths value is zero, which means that the number of path groups is unlimited. When you set the **-max** option to a value other than zero, path groups are created only for hierarchies with the maximum worst violations.

You can control path groups names by using the **-prefix** option. By default, the prefix is `synopsys_pg_`. For example, path group names by default would be `synopsys_pg_0`, `synopsys_pg_1`, and so on.

The generated **group_path** commands are listed in the output file specified by the **-file** option. You can edit the output file and add weight or **critical_range** options on particular path groups.

You can remove path groups generated by the **create_auto_path_groups** command by using the **remove_auto_path_groups** command.

Notes

If the design has inserted clock gates at the GTECH stage, you should run the **create_auto_path_groups** command only after the initial compile (with the **-mode mapped** option). This would lead to the creation of fewer path groups than running the **create_auto_path_groups** command before optimization (with the **-mode rtl** option), resulting in better runtime.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example creates path groups after design elaboration and removes them after compile. First, read the RTL, perform analyze and elaborate, and read the SDC constraints.

```
prompt> create_auto_path_groups -mode rtl # creates path groups at pre-compile stage
prompt> compile_ultra
prompt> report_qor                # reports both user and script-created path groups
prompt> remove_auto_path_groups  # removes only the script-created path groups
prompt> report_qor                # reports only the user-created path groups
```

The following example creates path groups after compile and removes them after incremental compile. First, read the RTL, perform analyze and elaborate, and read the SDC constraints.

```
prompt> compile_ultra
prompt> create_auto_path_groups -mode mapped # creates path groups at post-compile stage
prompt> compile_ultra -incremental
prompt> report_qor                # reports both user and script-created path groups
prompt> remove_auto_path_groups  # removes only the script-created path groups
prompt> report_qor                # reports only the user-created path groups
```

Consider an example with the following design hierarchy:

```
+-----+
|top      |
| +-----+ +-----+ |
| |m1  | |m2      | |
| +-----+ | +-----+ |
|          | |m3  | |
|          | +-----+ |
|          |          |
|          +-----+ |
|          |          |
+-----+
```

If paths groups are created after elaboration:

```
prompt> create_auto_path_groups -mode rtl -exclude {IO macro ICG}
INFO: Collecting hierarchies without optimize_registers...
INFO: Creating path groups for hierarchies without optimize_registers...
INFO: group_path -name synopsys_pg_0 -to [get_cells m1/*
-filter "is_hierarchical==false && is_sequential==true"] -priority 1
INFO: group_path -name synopsys_pg_1 -to [get_cells m2/*
-filter "is_hierarchical==false && is_sequential==true"] -priority 1
INFO: group_path -name synopsys_pg_2 -to [get_cells m2/m3/*
-filter "is_hierarchical==false && is_sequential==true"] -priority 1
```

If paths groups are removed after compile:

```
prompt> remove_auto_path_groups  
INFO:  remove_path_group synopsys_pg_2  
INFO:  remove_path_group synopsys_pg_1  
INFO:  remove_path_group synopsys_pg_0
```

If **set_optimize_registers** is set on m2 and paths groups are created after elaboration:

```
prompt> set_optimize_registers [get_designs [get_attribute [get_cells m2] ref_name]] true  
prompt> create_auto_path_groups -mode rtl -exclude {IO macro ICG}  
INFO:  Collecting hierarchies without optimize_registers...  
INFO:  Creating path groups for hierarchies without optimize_registers...  
INFO:  group_path -name synopsys_pg_0 -to [get_cells m1/* -filter "is_hierarchical==false &&  
is_sequential==true"] -priority 1
```

If paths groups are removed after compile:

```
prompt> remove_auto_path_groups  
INFO:  remove_path_group synopsys_pg_0
```

SEE ALSO

`remove_auto_path_groups(2)`

create_block_abstraction

Generates a block abstraction for the current design. The command identifies the interface logic of the current design and annotates the design in memory with the interface logic.

SYNTAX

```
status create_block_abstraction  
[-include objects]
```

Data Types

objects collection

ARGUMENTS

-include *objects*

Specifies the additional leaf cells and nets that are to be included in the block abstraction.

Logical hierarchical cells and pins are ignored.

If either one of the following conditions exist, the entire block is included in the block abstraction:

- The number of leaf cells included exceeds 95 percent of the total leaf cells in the block.
- The number of nets included exceeds 95 percent of the total nets in the block.

DESCRIPTION

This command creates a block abstraction of the current design and annotates the design in memory with the interface logic.

To save the block abstraction, you must use the **write_file** command immediately after creating the block abstraction. The **write_file** command writes the complete design, including the block abstraction information, into a single .ddc file.

Creating a block abstraction by using the **create_block_abstraction** command allows you to perform transparent interface optimization on the block in the top-level flow.

You can create block abstractions in either Design Compiler or Design Compiler in topographical mode. However, you must use topographical mode to optimize block abstractions with transparent interface optimization, and only block abstractions created in topographical mode are optimized.

At the top level, use the **set_top_implementation_options** command to specify which blocks should be integrated with the top-level

design as block abstractions.

The **create_block_abstraction** command first determines the interface logic.

The following netlist objects are part of the interface logic:

- All cells, pins, and nets in timing paths from input ports to registers or output ports.
- All cells, pins, and nets in timing paths to output ports from registers or input ports.
- The following clock-gating circuitry:
 - Any logic in the connection from the master clock to the generated clocks.
 - The clock trees that drive interface registers, including any logic in the clock tree.
 - The longest and shortest clock paths from the clock ports.
- The side-load cells of all nonideal and non-DRC disabled nets.

By default, the **create_block_abstraction** command ignores an input or inout port during interface logic identification, if the percentage of the total registers in the transitive fanout of the port is greater than or equal to the threshold percentage specified in the **abstraction_ignore_percentage** variable. (The default is 25.) The ports that are ignored are then connected to already-identified interface logic. Also, minimum and maximum critical timing paths are retained for these ignored ports.

This default helps to identify the test enable and reset ports of your design. Examine the current value of the **abstraction_ignore_percentage** variable and change it, if needed. The default might potentially ignore ports you do not want to ignore or fail to ignore ports that you do want to ignore. Carefully read the messages that the **create_block_abstraction** command issues when you use the default to see which ports have been ignored and to what percentage of registers they fanned out.

The **create_block_abstraction** command implicitly performs an **update_timing** on the design, if required.

The **create_block_abstraction** command ignores case analysis settings, if any, during interface logic identification. You must specify the appropriate case analysis settings at the top level. This is done to make the block abstraction more context independent.

The **create_block_abstraction** command assumes all latches found in the interface logic are potential borrowers; thus, all logic from I/O ports to flip-flops or output ports are identified as belonging to the interface logic.

Multicorner-Multimode Support

This command uses information from all scenarios.

For a multicorner-multimode design,

- The **create_block_abstraction** command automatically detects the presence of multiple scenarios (multiple modes or corners) and determines the interface logic for each scenario. The interface logic identified for each scenario is retained as the interface logic of the block abstraction. If an interface timing path is disabled in one scenario but enabled in another, the path is included in the interface logic.
- Net capacitance, resistance, and annotated delays are stored as part of abstraction information for each specified TLUPlus file and temperature.
- Power consumption of the design is calculated for each scenario. The calculated power data is stored in attributes in the design and can be used by the **report_power** command during the final assembly step of the entire chip.

EXAMPLES

The following example shows how to create and save a block abstraction:

```
prompt> read_ddc MY_DES.compiled.ddc  
prompt> create_block_abstraction  
prompt> write_file -hierarchy -format ddc -output MY_DES.abstract.ddc
```

SEE ALSO

- abstraction_ignore_percentage(3)
- check_block_abstraction(2)
- report_block_abstraction(2)
- set_top_implementation_options(2)

create_bounds

Creates a fixed move bound or floating group bound in the design.

SYNTAX

```
status create_bounds
  [-name bound_name]
  [-coordinate {llx1 lly1 urx1 ury1 ...}]
  [-dimension bound_dimension]
  [-diamond central_object]
  [-effort low | medium | high | ultra]
  [-type soft | hard]
  [-exclusive]
  [-color color]
  [-cycle_color]
  [-repelling diamond | rect]
  object_list
```

Data Types

<i>bound_name</i>	string
<i>llx1</i>	float
<i>lly1</i>	float
<i>urx1</i>	float
<i>ury1</i>	float
<i>bound_dimension</i>	list
<i>central_object</i>	object
<i>color</i>	integer or string
<i>object_list</i>	list

ARGUMENTS

-name *bound_name*

Specifies the name of the bound.

-coordinate {*llx1 lly1 urx1 ury1 ...*}

Specifies rectangular move bounds by specifying the coordinates of the lower-left and upper-right corners for each rectangle. The coordinates are in microns relative to the chip origin.

Each combination of {*llx lly urx ury*} defines a target placement area for the objects. The coarse placement engine can place cells in any of the rectangles. These placeable areas can overlap or be disjoint.

You can specify rectilinear move bounds by dividing the rectilinear region into individual rectangular bounds and specifying the

rectangles in the coordinate list.

-dimension *bound_dimension*

Specifies the dimension of a group bound or diamond bound in microns. The dimension of a group bound is specified as {*width height*}, while the dimension of a diamond bound is specified as *extent*.

-diamond *central_object*

Creates a diamond bound centered on the specified object, which can be a port, cell, or pin.

The objects in the bound are constrained to lie within the distance specified by the **-dimension** option (measured as a Manhattan distance) of the specified object. A diamond bound is always a soft bound.

If you specify this option, you must also specify the **-dimension** option.

-effort *low | medium | high | ultra*

Specifies the effort to bring cells closer inside a group bound.

The default is **medium**.

This option is mutually exclusive with the **-coordinate** and **-dimension** options.

-type *soft | hard*

Specifies the type of the bound to be either **hard** or **soft**.

To use this option, you must also specify either the **-coordinate** or **-dimension** option. The bound type cannot be set for automatically generated group bounds or diamond bounds.

The default is **soft**.

-exclusive

Creates an exclusive move bound.

Exclusive move bounds require all of their cells to be placed inside them and prohibit the placement of other cells in the same area. Exclusive move bounds are respected by both coarse placement and legalization.

-color *color*

Specifies the move bound color. You can specify the color either by specifying an integer value between 0 and 63 or by specifying one of the following keywords: **black**, **blue**, **green**, **cyan**, **brown**, **purple**, **red**, **magenta**, **salmon**, **orange**, **yellow**, or **white**.

The default is **no_color**, meaning that no color is specified.

-cycle_color

Allows the tool to automatically assign a move bound color.

By default, this option is **off**.

-repelling *diamond | rect*

Specifies a bound to be repelling rather than attractive. A repelling bound specifies a floating area between a pair of objects to keep them apart in the bound. You can specify either a diamond-shaped bound (**diamond**) or a rectangle-shaped bound (**rect**) for specifying the shape of the repelling bound.

object_list

Specifies a list of cells, ports, and pins to be included in the bound.

- If a cell is a hierarchical cell, the bound applies to all cells in the subdesign.
- If a port is a hierarchical port, the corresponding port is attached to the bound instead.
- If a pin is a leaf pin, the lower-left point of the first geometry of the pin is marked on the cell of the pin, and the cell is added to the bound.
- If a pin is a hierarchical pin, it is ignored.

A cell can be assigned to only one move bound; an error occurs if a specified cell already belongs to another move bound.

This argument is optional; you can create an empty move bound and later associate cells with it by using the **update_bounds** command.

DESCRIPTION

The **create_bounds** command allows you to define region-based placement constraints for coarse placement.

There are three types of bounds: move bounds, group bounds, and diamond bounds.

- Move bounds restrict the placement of cells to a specific region of the core area.

To create a move bound, use the **-coordinate** option.

- Group bounds are floating region constraints. Cells in the same group bound are placed within a specified bound, but the absolute coordinates are not fixed. Instead, they are optimized by the placer.

To create a group bound, use the **-dimension** option.

- Diamond bounds are region constraints centered on a specific object, which can be fixed or floating. Other objects in the same diamond bound are placed within the specified Manhattan distance from the central object but their absolute coordinates are not fixed. Instead, they are optimized by the placer.

To create a diamond bound, use the **-diamond** option together with the **-dimension** option.

If you do not use any of these options, the tool creates a group bound with a bounding box computed internally by the tool. In this case, you can use the **-effort** option to specify the effort level used to bring the cells closer. All automatically generated bounds are soft bounds; you cannot use the **-type** option to change the bound type for these group bounds.

Generally, there is no guarantee that cells are placed completely within the bounds. For instance, coarse placement can violate the bounds if the quality of its primary placement objectives would otherwise be destroyed.

In these situations, you should revisit the bounds and floorplan to ensure that the design is not overconstrained. Alternatively, you can use the **-type hard** option to specify the bound type to be **hard**. The default is **soft**. The coarse placer tries to honor the hard bound as hard constraints while sacrificing other objectives, such as timing and routability. You should not use many hard bounds because this can lead to inferior placement solutions.

The bounds created by the **create_bounds** command are persistent and do not need to be re-created when the design is reloaded.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example constrains the placement of the INST_1 instance to lie within the rectangle with its lower-left corner at (100, 100) and its upper-right corner at (200, 200):

```
prompt> create_bounds -name "temp" -coordinate {100 100 200 200} INST_1
```

The following example creates a hard rectilinear move bound on a hierarchical instance named INST2. The rectilinear move bound is created as a set of two rectangles: one with its lower-left corner at (10, 10) and its upper-right corner at (30, 20), and one with its lower-left corner at (20, 20) and its upper-right corner at (30, 30).

```
prompt> create_bounds -name "temp2" \  
-coordinate {10 10 30 20 20 20 30 30} \  
-type hard INST_2
```

The following example creates a diamond bound centered on the pin MOD/U1/A. The cells MOD/INST1 and MOD/INST2 are constrained to lie at most 64 microns of Manhattan distance away from MOD/U1/A.

```
prompt> create_bounds -name "temp3" \  
-dimension 64 \  
-diamond MOD/U1/A \  
{MOD/INST1 MOD/INST2}
```

SEE ALSO

```
remove_bounds(2)  
report_bounds(2)  
set_cell_location(2)  
update_bounds(2)
```

create_bsd_patterns

Generates a set of functional patterns for a boundary-scan design.

SYNTAX

```
status create_bsd_patterns
  -output test_program_name
  [-effort low | medium | high]
  [-type pattern_type]
  [-setup_instructions instruction_list]
  [-bsd_test_setup enable | disable]
  [-jtag_reset enable | disable]
```

Data Types

```
test_program_name  string
pattern_type       string
instruction_list    list
```

ARGUMENTS

-output *test_program_name*

Specifies the name of the output test program. On completion, **create_bsd_patterns** writes out the Synopsys pattern database (*test_program_name.vdb*) file that contains the generated patterns and the fault status information. By default, the *test_program_name* is *design_name* and the file is named *design_name.vdb*. This file is placed in the current working directory.

-effort low | medium | high

Controls the effort used to search for implemented instructions if the boundary-scan database has not been previously built by the **check_bsd**, **insert_dft**, or **read_bsd** command and the tool must infer the implemented instructions. For more information, see the man page for the **check_bsd** command.

-type *pattern_type*

Generates vectors to test the various aspects of the boundary scan of the design.

This option allows you to generate vectors to test bsr, tap, tdr, and reset mechanisms of the boundary scan of the design separately. This option allows generation of leakage vectors and all functional vectors (bsr, tap, tdr, and reset vectors). The option also allows generation of vectors to test AC receivers and transmitters.

You must specify one of the following values:

all generates everything (the default).

functional generates {tap_controller, reset, bsr, tdr, ac_input_pulse, ac_input_train, ac_output_pulse, ac_output_train}.

dc_parametric generates vectors to {bsr, leakage}. Can also be used to measure voltage and current levels for I/O.

tap_controller generates vectors to test the tap finite state machine function.

reset generates vectors to test the asynchronous and synchronous reset mechanism.

tdr generates vectors to test that each instruction implemented selects the corresponding test data register.

bsr generates vectors to test the function of the boundary scan register.

leakage generates vectors to test I/O leakage.

ac_input_pulse generates vectors for AC input tests with EXTEST_PULSE instruction. These vectors test the preset/transition/single ended/dc behaviors of AC receivers and the DC behavior of the AC instruction.

ac_input_train generates vectors for AC input tests with EXTEST_TRAIN instruction. These vectors test the preset/transition/single ended/dc behaviors of AC receivers and the DC behavior of the AC instruction.

ac_output_pulse generates vectors for AC output tests with EXTEST_PULSE instruction. These vectors test the transition behavior of AC drivers with the AC instruction. The vectors also test AC/DC selection cells.

ac_output_train generates vectors for AC output tests with EXTEST_TRAIN instruction. These vectors test the transition behavior of AC drivers with the AC instruction. The vectors also test AC/DC selection cells.

The default behavior is to generate both functional and DC parametric vectors. This is the same as running the command with the **-type all** option.

-setup_instructions *instruction_list*

Specifies one or more user-defined instructions whose associated test data registers are to be initialized at the beginning of the test program.

This method creates a test_setup procedure for the test program that initializes the user-defined test data registers (UTDRs) associated with the specified instructions. The initialization data is specified with the **-bsd_init_data** option of the **set_bsd_instruction** command.

By default, no test_setup procedure is created. This option is mutually exclusive with the **-bsd_test_setup** option.

-bsd_test_setup enable | disable

Specifies that a custom test_setup procedure to be used for the test patterns.

This method inserts the test_setup section from the provided STIL protocol file into the created patterns. The setup protocol must be read using the **read_test_protocol -section test_setup** command prior to pattern creation.

By default, no test_setup procedure is created. This option is mutually exclusive with the **-setup_instructions** option.

-jtag_reset enable | disable

Specifies whether to enable JTAG reset operations during the test program.

If you are using the **-setup_instructions** or **-bsd_test_setup** option to initialize configuration registers at the beginning of the test program, and the configuration registers are affected by the JTAG reset signal, you can use the **-jtag_reset disable** option to suppress JTAG resets during the test program.

When the **-setup_instructions** option is used, BSD Compiler creates a test_setup procedure that asserts the JTAG reset signal before loading the specified configuration registers, but it suppresses all subsequent JTAG reset signals in the test program.

When the **-bsd_test_setup** option is used, you supply a custom test_setup procedure that is responsible for asserting JTAG reset and initializing the configuration registers. BSD Compiler suppresses all subsequent JTAG reset signals in the test program.

If the **tap_controller** or **reset** type is specified with the **-type** option, the **-jtag_reset disable** option is ignored because the JTAG

reset signal must be asserted during these tests. If the **all** type is specified, the **tap_controller** and **reset** tests are omitted from the test program.

This option can only be specified together with the **-setup_instructions** or **-bsd_test_setup** option.

By default, JTAG resets are enabled.

DESCRIPTION

The **create_bsd_patterns** command generates functional test patterns for a boundary-scan design. If the boundary-scan database was not previously built by the **check_bsd**, **insert_dft**, or **read_bsd** command, the command performs an implicit compliance check before proceeding. If a boundary-scan design is non-compliant, the **create_bsd_patterns** command does not generate any functional vectors. If a boundary-scan design is compliant, the **create_bsd_patterns** command generates functional test patterns to test the synchronous and asynchronous TAP reset, the TAP controller finite state machine, the implemented boundary-scan instructions and associated test data registers, and the boundary-scan register.

If the design contains IEEE 1149.6 logic, the generated functional vectors also include tests for AC receivers and transmitters. Note that these vectors are not generated if compliance checker is run prior to pattern generation.

The test patterns are stored in the Core Test Language (CTL) model for the design, which can be converted to different formats using the **write_test** command.

EXAMPLES

The following example performs functional testbench generation on the current design and writes out the bscan.vdb Synopsys pattern database:

```
prompt> create_bsd_patterns -out bscan
```

```
Loading db file '/u/root_dir/libraries/my_class.db'
Loading design 'M1'
...Starting IEEE 1149.1 Compliance Checking.
...Inferring the boundary scan protocol for the design 'M1'.
...Generating the BSD test patterns for the design 'M1'.
.....Generating vectors to test the asynchronous test logic reset.
.....Generating vectors to test the synchronizing sequence of 5 1's on tms.
.....Generating vectors to test the TAP FSM.
.....Generating vectors to test boundary scan instructions.
.....Generating vectors to test the 'BYPASS' instruction.
.....Generating vectors to test the 'EXTEST' instruction.
.....Generating vectors to test the 'SAMPLE' instruction.
.....Generating vectors to test the 'CLAMP' instruction.
.....Generating vectors to test the 'HIGHZ' instruction.
.....Generating vectors to test the 'IDCODE' instruction.
.....Generating vectors to test the boundary scan register.
...Writing test program bscan to file /tmp/bsdvec/bscan.vdb.
```

BSD test patterns are generated successfully.

1

SEE ALSO

check_bsd(2)
set_bsd_instruction(2)
write(2)
write_test(2)

create_buffer_tree

Creates a buffer tree for the specified driver pins and nets.

SYNTAX

```
status create_buffer_tree  
  [-from pin_net_list]  
  [-incremental]  
  [-no_legalize]  
  [-on_route [-skip_detail_route]]  
  [-align_hierarchy_for_long_nets]
```

Data Types

pin_net_list collection

ARGUMENTS

-from *pin_net_list*

Specifies the driver pins and nets for which the tool creates buffer trees. When the **-from** option is not specified, the **create_buffer_tree** command works on all drivers with a transitive fanout of 5 or more.

-incremental

Sets the scope of creating the buffer tree for the specified nets rather than the entire buffer tree driven by the specified nets. When this option is specified, the **create_buffer_tree** command constructs a buffer tree, if needed, on each net specified by the **-from** option to reduce the high fanout of each net, but it does not remove any existing buffers or inverters. So, the scope of creating buffer trees is a single specified net.

-no_legalize

Disables placement legalization at the end of buffering. By default Design Compiler does not do placement legalization so the option doesn't have any effect.

-on_route

Perform the postroute optimization flow, where it fixes DRC violations by using the `route_opt` command and then the `focal_opt` command. It focuses mainly on fixing maximum net length, so you need to specify the `max_net_length` constraint by using the `set_max_net_length` command, before using this option. The final database is a detail routed database.

It is recommended to use the **-align_hierarchy_for_long_nets** option with the **-on_route** option to ensure that optimization occurs for long nets crossing logical hierarchies.

You can use the **-from** option with this option to fix DRC violations for the concentrated nets. However, if you use the **-from** option, the design must be detail routed and you cannot use the **-skip_detail_route** or **-align_hierarchy_for_long_nets** options.

This option is not supported in Design Compiler topographical mode and will be ignored.

-skip_detail_route

Skips detail routing when used with the **-on_route** option. By default the **-on_route** option triggers full detail routing after topology-based DRC optimization. This option can reduce runtime of the **on_route** option by skipping the final detail route.

This option can only be used with the **-on_route** option. This option cannot be combined with the **-from** option.

This option is not supported in Design Compiler topographical mode and will be ignored.

-align_hierarchy_for_long_nets

Enables additional optimizations and port punching for long nets that crosses multiple hierarchies. Long nets are those nets that violate the maximum net length constraint for the design. You can specify `max_net_length` constraint via command `set_max_net_length` before using this option. If you use this option and the design does not have a maximum net length constraint, the command fails with an error message.

When you use this option, the tool first performs high-fanout net fixing and also fixing DRC violations, such as maximum transition and capacitance, for low fanout nets. It then performs preroute topology-based buffering and port punching for the nets that violate the maximum net length constraint and crosses multiple hierarchies. If both of these conditions are not met, this additional optimization is not performed.

This option does not fix all maximum net length constraint. You have to rely on the **-on_route** option to fix the remaining violations.

This option can be used alone or with the **-on_route** option, but it cannot be used with the **-from** options.

This option is not supported in Design Compiler topographical mode and will be ignored.

DESCRIPTION

The **create_buffer_tree** command creates a buffer tree for each specified driver pin and for the driver pin of each specified net.

The **create_buffer_tree** command generates a hierarchical buffer tree, and buffers are inserted across hierarchical boundaries. The command is location-based. Therefore, the specified driver pin should not be a hierarchical pin because a hierarchical pin does not have a location.

Use this command on a placed design only. The command requires the following libraries and information:

- The physical libraries that correspond to the specified logic libraries.
- The capacitance and resistance per unit length derived from the physical library or user-specified.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following example creates a buffer tree that is implemented at driver a/O:

```
prompt> create_buffer_tree -from [get_pins a/O]
```

The following example creates buffer trees for driver a/O and net n1:

```
prompt> create_buffer_tree -from {[get_pins a/O] [get_nets n1]}
```

SEE ALSO

- `remove_buffer_tree(2)`
- `report_buffer_tree(2)`
- `report_ahfs_options(2)`
- `report_net_fanout(2)`
- `set_ahfs_options(2)`

create_bus

Creates a port bus or a net bus.

SYNTAX

```
status create_bus  
  object_list  
  bus_name  
  [-type type_name]  
  [-sort]  
  [-no_sort]  
  [-start start_bit]  
  [-end end_bit]
```

Data Types

<i>object_list</i>	list
<i>bus_name</i>	string
<i>type_name</i>	string
<i>start_bit</i>	integer
<i>end_bit</i>	integer

ARGUMENTS

object_list

Specifies a list of ports or nets to be put into a bus. If both ports and nets have the same names, the ports are put into the bus.

This option is required.

bus_name

Specifies the name of the bus. This name cannot be the same as any other bus or object of the same type. Port bus names must be different from the names of ports, and net bus names must be different from the names of nets.

This option is required.

-type *type_name*

Assigns the specified *type_name* to the bused port. Valid types are port or net. This name appears in port declarations when a design with the bused ports is written to a file in the VHDL format. All buses that are assigned the same type name must be the same width (contain the same number of members).

-sort

Sorts the bits of the bus specified in *object_list* in alphanumeric order. By default, bits sort in reverse alphanumeric order.

-no_sort

Specifies that the order of the bits in the bus should be the same as specified on the command line. By default, bits sort in reverse alphanumeric order.

-start *start_bit*

Specifies the start bit for the bus.

-end *end_bit*

Specifies the end bit for the bus.

DESCRIPTION

The **create_bus** command creates a bus object of the type port or net. The number of objects in the list determines the bus width. Buses appear as multibit ports on a design or as multiwire nets in a design. This command groups any number of ports or any number of nets into a bus object. Unless the **-sort** or **-no_sort** option is selected, the specified objects are sorted by reverse alphanumeric order of names before the bus is created.

When using Design Vision to create a design schematic, bused nets are inferred from bused ports or pins in the design. Bused nets only appear as nets in a schematic if they are connected to a bused port.

If the start bit and end bit for the bus are specified with the **-start** and **-end** options, the bus is created with these start and end indices. If only the start bit is specified, an upward-going bus starting at that start bit is built. If only the end bit is specified, an upward-going bus ending at that end bit is built.

EXAMPLES

This example groups the A1, A2, and A3 ports into a bus named *A*. The ports must already exist. The order in which the ports appear in the bus is A3, A2, A1.

```
prompt> create_bus {A1 A2 A3} A
```

This example groups the B1, B2, and B3 nets into a bus named *B*. The nets must already exist and there must not be any ports with the same names. The order in which the nets appear in the bus is B3, B2, B1.

```
prompt> create_bus {B1 B2 B3} B
```

This example groups the C1, C2, and C3 nets into a bus named *C*. The nets must already exist and because the object *net* is defined in the **get_nets** command, it does not matter if there are ports with the same name. The order in which the nets appear in the bus is C3, C2, C1.

```
prompt> create_bus [get_nets C1 C2 C3] C
```

This example groups the existing D1, D2, and D3 ports into a bus named *D* and assigns the 3-bit type to the bus:

```
prompt> create_bus {D1 D2 D3} D -type "3-bit"
```

This example groups the existing D1, D2, and D3 ports into a bus named *D* and assigns it the index range 6-to-4. Port D1 is assigned index 6, port D2 to index 5, and port D3 to index 4.

```
prompt> create_bus {D1 D2 D3} D -start 6 -end 4
```

This example groups the E1, E2, and GH ports into a bus named *E*. The ports must already exist. Because the **-sort** option is used, the first bit is port E1, the second bit is port E2, and the third bit is port GH.

```
prompt> create_bus {E2 GH E1} E -sort
```

This example groups the R1, R2, and GH ports into a bus named *R*. The ports must already exist. Because the **-no_sort** option is used, the first bit is port R1, the second bit is port R2, and the third bit is port GH.

```
prompt> create_bus {R1 R2 GH} R -no_sort
```

This example groups the ports in the MID subdesign into a bus named *NEW*. U1 is a unique instantiation of MID.

```
prompt> create_bus {U1/R1 U1/R2 U1/GH} NEW -no_sort
```

This example groups the nets in the MID subdesign into a bus named *NET_BUS*. U1 is a unique instantiation of MID.

```
prompt> create_bus [get_nets U1/R1 U1/R2 U1/GH] NET_BUS -no_sort
```

SEE ALSO

- get_nets(2)
- remove_bus(2)
- report_bus(2)

create_cache

Populates the cache directories with instances of the requested synthetic modules.

SYNTAX

```
status create_cache
  -module module_list
  [-implementation implementation_list]
  [-parameters parameter_list]
  [-operating_condition operating_condition]
  [-wire_load list]
  [-report]
```

Data Types

```
module_list      list
implementation_list list
parameter_list   list
operating_condition string
list             list
```

ARGUMENTS

-module *module_list*

Lists the synthetic module names used to create and place in the cache. Strings in the list can contain the asterisk (*) wildcard character (for example: { DW_01 DW_* *_adders }).

-implementation *implementation_list*

Restricts the part creation to the implementations in the list. Strings in the list can contain the asterisk (*) wildcard character.

-parameters *parameter_list*

Restricts the part creation to instances of the parts that have the parameters in the list. The **create_cache** command displays warning or error messages when an incorrect number of parameters is specified; a list of the required parameters is also displayed. This argument is required unless the synthetic module cannot be parameterized. For more information on *parameter_list*, see the EXAMPLES section.

-operating_condition *operating_condition*

Optimizes and models the created parts using the specified *operating_condition*. If not specified, the default operating condition in the target library is used.

-wire_load *list*

Lists the wire load models to use to optimize and model the created parts. If no wire load is provided, the default wire load model in the target library is used.

-report

Generates a brief timing report that displays the longest path in the created part.

DESCRIPTION

The **create_cache** command creates instances of the specified synthetic modules and places them in the cache. If no implementation name is given, an instance of all valid implementations of the synthetic module is created.

See the DesignWare documentation for further information.

EXAMPLES

The following example shows the use of the **create_cache** command:

```
prompt> create_cache -module {DW01_add} \
  -implementation {cla_rpl} \
  -parameters { "width = 8" } \
  -operating_condition "Best_Case_Commercial"
```

The **-parameters** command argument has some additional features. The *parameter_list* is a space-separated list of quoted parameter specifications (for example: {"N=8,M=6" "N=3"}). Each quoted parameter specification is a comma-separated list of parameter settings. Parameter settings are a parameter name, followed by an equal sign (=), followed by a parameter value. Parameter settings can also support ranges for the parameter value "width= [8;17]", where a square bracket ("[") means the end point is inclusive, and a parenthesis ("]") means the end point is exclusive: Thus, **-parameters {"width = [8;17]"}** means create instances of the synthetic modules with parameter "width" set to 8 to 16.

The following example creates DW02 multipliers for all possible combinations of the A_width and B_width parameters in the given range - 12 to 16 X 8 to 16. The second set of parameters requests the creation of a 32x32 bit multiplier. Since no implementation name is provided, an instance of all valid implementations of the DW02_mult module is created.

```
prompt> create_cache -module DW02_mult \
  -wire_load {"70x70" "80x80"} \
  -p {"A_width=(11;16], B_width=(7;16]", \
  "A_width=32, B_width=32" }
```

SEE ALSO

remove_cache(2)
report_cache(2)
cache_read(3)
cache_write(3)

create_cell

Creates leaf or hierarchical cells in the current design or its subdesigns.

SYNTAX

```
status create_cell
      cell_list
      [reference_name]
      [-hierarchical]
      [-logic 0 | 1]
      [-only_physical]
```

Data Types

```
cell_list      list
reference_name string
```

ARGUMENTS

cell_list

Specifies the names of cells created in the current design. Each cell name must be unique within the current design.

reference_name

Specifies the design or library cell that new cells reference. You must specify the *reference_name* unless you use the **-logic** option. Ports on the reference determine the name, number, and direction of pins on the new cell.

-hierarchical

Creates hierarchical cell instances and designs with the name given by *cell_list* if the *reference_name* is not specified.

If you specify the *reference_name*, the *cell_list* must have a single element. The command creates the hierarchical cell instance with the name given by the single *cell_list* and also creates the design with the name given by *reference_name*.

-logic 0 | 1

Specifies that the new cell generates a logic 0 or logic 1 value. The logic value must be either **0** or **1**. By using this option, the cell contains a single output pin. By default, this option is off.

-only_physical

Creates a new physical or physical-only cell using a reference from the physical library. The **-only_physical** option sets the `is_physical_only` attribute on the created physical-only cell. After you create a physical-only cell, you must assign a location to that cell. Unlike physical cells with logic functions, the tool does not assign a location to a physical-only cell during synthesis. To assign a location, use the **set_cell_location** command, Tcl commands, or a DEF file. By default, this option is off.

DESCRIPTION

This command creates new leaf or hierarchical cells in the current design or its subdesigns based on the *cell_list* argument. New leaf cells are the instantiation of an existing design or library cell. New hierarchical cells are the instantiation of a new design. New cells are the instantiation of an existing design, a library cell, a logic 0 generator, or a logic 1 generator.

The created cells are unplaced. To be viewed properly in the GUI, the cells must be placed either manually by using the **set_cell_location** command or automatically by using placement commands.

To remove cells from the current design, use the **remove_cell** command.

Although the *reference_name* argument accepts names in the format *library/library_cell*, the command might not instantiate the actual library cell from the specified library. The actual library cell to be used is determined by the current link library settings.

EXAMPLES

The following example creates a cell named *cell1* under the subdesign corresponding to the *mid1* cell:

```
prompt> create_cell {mid1/cell1} my_lib/AND2
Creating cell 'cell1' in design 'mid'.
```

The following example creates a cell named *cell2* in the design corresponding to the *m1/U1* and *m2/U1* instances. The cell names *cell2* is a logic 0 generator. The design corresponding to *m1/U1* and *m2/U1* must be unique.

```
prompt> create_cell {m1/U1/cell2 m2/U1/cell2} -logic 0
```

The following example creates cells using existing designs as references:

```
prompt> create_design ALPHA

prompt> current_design ALPHA

prompt> create_cell U1 ADDER

prompt> create_cell U2 SUBTRACTOR
```

The following example creates logic 1 and logic 0 cells:

```
prompt> create_cell LOGIC_ONE -logic 1

prompt> create_cell LOGIC_ZERO -logic 0
```

The following example creates leaf cells using library cells as references.

```
prompt> create_cell {U3 U4} my_lib/NAND2

prompt> create_cell "U5" my_lib/NOR2
```

The following example creates hierarchical cell *H1* with the new design name *DESIGN1*, and creates leaf cell *U2* under the new hierarchical cell using library cells as references:

```
prompt> create_cell -hierarchical H1 DESIGN1
```

```
prompt> create_cell H1/U2 my_lib/NAND2
```

The following example creates hierarchical cells *H2* and *H3* with the design name *H2* and *H3* separately. Also, it creates leaf cell *U3* under the new hierarchical cell *H2* using library cells as references.

```
prompt> create_cell -hierarchical {H2 H3}
```

```
prompt> create_cell H2/U3 my_lib/NAND2
```

The following example creates physical cells using physical library cells as references:

```
prompt> create_cell -only_physical {U3 U4} lib.pdb:my_lib/pad0
```

```
prompt> create_cell -only_physical phys_inst1 test_lib.pdb:mylib/PAD0
```

SEE ALSO

- current_design(2)
- remove_cell(2)
- report_cell(2)
- set_cell_location(2)

create_clock

Creates a clock object and defines its waveform in the current design.

SYNTAX

```
status create_clock
  [-name clock_name]
  [-add]
  [source_objects]
  [-period period_value]
  [-waveform edge_list]
  [-comment comment_string]
```

Data Types

```
clock_name    string
source_objects list
period_value  float
edge_list     list
comment_string string
```

ARGUMENTS

-name *clock_name*

Specifies the name of the clock being created. If you do not use this option, the clock is given the same name as the first clock source specified in *source_objects*. If you do not use *source_objects*, you must use this option, which creates a virtual clock not associated with a port or pin. Use this option along with *source_objects* to give the clock a more descriptive name than that of the pin or port where it is applied.

If you specify the **-add** option, you must use the **-name** option and the clocks with the same source must have different names.

-add

Specifies whether to add this clock to the existing clock or to overwrite the existing clock. Use this option to capture the case where multiple clocks must be specified on the same source for simultaneous analysis with different clock waveforms. When you specify this option, you must also use the **-name** option. Defining multiple clocks on the same source pin or port causes longer runtime and higher memory usage than a single clock, because the synthesis timing engine must explore all possible combinations of launch and capture clocks. Use the **set_false_path** command to disable unwanted clock combinations. This option is ignored (the default), unless multiple clocks analysis is enabled by setting the **timing_enable_multiple_clocks_per_reg** variable to **true**.

source_objects

Specifies a list of pins or ports on which to apply this clock. If you do not use this option, you must use **-name *clock_name***, which

creates a virtual clock not associated with a port or pin. If you specify a clock on a pin that already has a clock, the new clock replaces the old clock unless you use the **-add** option.

-period *period_value*

Specifies the period of the clock waveform in library time units.

-waveform *edge_list*

Specifies the rise and fall edge times, in library time units, of the clock over an entire clock period. The first time in the list is a rising transition, typically the first rising transition after time zero. There must be an even number of increasing times, and they are assumed to be alternating rise and fall times. The numbers must represent one full clock period. If **-waveform *edge_list*** is not specified, but **-period *period_value*** is, a default waveform with a rise edge of 0.0 and a fall edge of *period_value*/2 is assumed.

-comment *comment_string*

Allows the command to accept a comment string. The tool honors the annotation and preserves it with the SDC object so that the exact string is written out when the constraint is written out when you use the **write_sdc** or **write_script** command. The comment remains intact through the synthesis, place-and-route, and timing-analysis flows.

DESCRIPTION

The **create_clock** command creates a clock object in the current design. The command defines the specified *source_objects* as clock sources in the current design. A pin or port can be a source for a single clock. If *source_objects* is not specified, but a *clock_name* is given, a virtual clock is created. A virtual clock can be created to represent an off-chip clock for input or output delay specification. For more information about input and output delay, refer to the **set_input_delay** and **set_output_delay** command man pages.

Clock objects hold attributes that affect the clock network, such as **dont_touch_network**, **fix_hold**, and **propagated_clock**. Using **create_clock** on an existing clock object overwrites the attributes previously set on the clock object. The **create_clock** command also defines the waveform for the clock. The clock can have multiple pulses per period. Setup and hold path delays are automatically derived from the clock waveforms of the path startpoint and endpoint. The **fix_hold** attribute (set by the **set_fix_hold** command) directs **compile** to fix hold violations for a clock.

By default, a new path group is created for the clock. This groups together the endpoints related to this clock for cost function calculation. To remove the clock from its assigned group, use the **group_path** command to reassign the clock to another group or to the default path group. For more information, refer to the **group_path** command man page.

The new clock has ideal timing, so no propagation delay through the clock network is assumed. To enable propagation delay through the clock network, use the **set_propagated_clock** command. To add skew or uncertainty to the ideal waveform, use the **set_clock_latency** or **set_clock_uncertainty** command.

To show information about all clock sources in a design, use the **report_clock** command. To get a list of clock sources, use the **get_clocks** command. To return sequential cells related to a given clock, use the **all_registers** command. To undo **create_clock**, use the **remove_clock** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example creates a clock on port *PHI1* with a period of 10.0, rise at 5.0, and fall at 9.5:

```
prompt> create_clock "PHI1" -period 10 -waveform {5.0 9.5}
```

In the following example, the clock has a falling edge at 5 and a rising edge at 10, with a period of 10. Because the **-waveform** option expects the edges to be ordered first rise then fall, and to increase in value, the fall edge can be given as 15; that is, the next falling edge after the first rise edge at 10.

```
prompt> create_clock "PHI2" -period 10 -waveform {10 15}
```

The following example creates a clock named *CLK* on pin *u13/Z*, with a period of 25, fall at 0.0, rise at 5.0, fall at 10.0, rise at 15.0, and so forth:

```
prompt> create_clock "u13/Z" -name "CLK" -period 25 -waveform {5 10 15 25}
```

The following example creates a virtual clock named *H12* with a period of 10.0, rise at 0.0, and fall at 5.0:

```
prompt> create_clock -name "PHI2" -period 10 -waveform {0.0 5.0}
```

The following example creates a clock with multiple sources and a complex waveform:

```
prompt> create_clock -name "clk2" -period 10 -waveform {0.0 2.0 4.0 6.0} \  
{clkgen1/Z clkgen2/Z clkgen3/Z}
```

SEE ALSO

- `all_clocks(2)`
- `all_registers(2)`
- `check_timing(2)`
- `compile(2)`
- `current_design(2)`
- `get_clocks(2)`
- `group_path(2)`
- `remove_clock(2)`
- `report_clock(2)`
- `reset_design(2)`
- `set_clock_latency(2)`
- `set_clock_uncertainty(2)`
- `set_dont_touch_network(2)`
- `set_fix_hold(2)`
- `set_max_delay(2)`
- `set_min_delay(2)`
- `set_output_delay(2)`
- `set_propagated_clock(2)`

create_command_group

Creates a new command group.

SYNTAX

```
string create_command_group  
[-info info_text]  
group_name
```

ARGUMENTS

-info *info_text*

Help string for the group

group_name

Specifies the name of the new group.

DESCRIPTION

The **create_command_group** command is used to create a new command group, which you can use to separate related user-defined procedures into functional units for the online help facility. When a procedure is created, it is placed in the "Procedures" command group. With the **define_proc_attributes** command, you can move the procedure into the group you created.

The *group_name* can contain any characters, including spaces, as long as it is appropriately quoted. If *group_name* already exists, **create_command_group** quietly ignores the command. The result of **create_command_group** is always an empty string.

EXAMPLES

The following example demonstrates the use of the **create_command_group** command:

```
prompt> create_command_group {My Procedures} -info "Useful utilities"
```

```
prompt> proc plus {a b} { return [expr $a + $b] }
```

```
prompt> define_proc_attributes plus -command_group "My Procedures"
```



```
prompt> help
```

```
My Procedures:
```

```
plus
```

```
...
```

SEE ALSO

`define_proc_attributes(2)`

`help(2)`

`proc(2)`

create_core_area

Defines the core area for the current design.

SYNTAX

```
status create_core_area
  -coordinate {{llx lly} {urx ury}}
  [-tile_name tile_name]
  [-direction horizontal | vertical]
  [-name core_name]
```

Data Types

<i>llx</i>	integer
<i>lly</i>	integer
<i>urx</i>	integer
<i>ury</i>	integer
<i>tile_name</i>	string
<i>core_name</i>	string

ARGUMENTS

-coordinate {{*llx lly*} {*urx ury*}}

Specifies a rectangular core area based on the specified lower-left and upper-right coordinates.

-tile_name *tile_name*

Specifies a unit tile for the core area.

-direction horizontal | vertical

Specifies the direction for the tiles. Tiles are added in rows to create the core area.

-name *core_name*

Specifies the core name. Design Compiler topographical mode does not support this option. The tool ignores any name specified by this option.

DESCRIPTION

The **create_core_area** command is deprecated. Use the **create_site_row** command to create a core area.

The **create_core_area** command defines the core area for the current design. The command cannot overwrite any previously-specified core area. The tool issues a warning for this condition.

EXAMPLES

The following example shows how to create a core area:

```
prompt> create_core_area -coordinate {{0 0} {200 200}}
```

SEE ALSO

create_site_row(2)
remove_core_area(2)

create_design

Creates a design in dc_shell memory.

SYNTAX

```
status create_design
      design_name
      [file_name]
```

Data Types

```
design_name  string
file_name   string
```

ARGUMENTS

design_name

Specifies the name of the design to be created. A *design_name* must be unique within the default file unless *file_name* is specified.

file_name

Specifies the name of the file in which the new design is placed. The *file_name* argument is optional. By default, the design is placed in *design_name.db* in the current directory. The file is not written by **create_design**. The design is placed in a default file in dc_shell.

DESCRIPTION

The *create_design* command creates a new design in dc_shell. The new design created is empty. A new design contains no sub-objects. Sub-objects can be created with the **create_bus**, **create_cell**, **create_net**, and **create_port** commands. To make the new design the working design in dc_shell, use the **current_design** command. To remove designs from dc_shell, use the **remove_design** command.

EXAMPLES

The following example uses **create_design** to create a design:

```
prompt> create_design ADDER
```

```
prompt> list_designs
```

```
prompt> current_design ADDER
```

The following example specifies the default file for the new design:

```
prompt> create_design SUBTRACTOR "/tmp/blat.db"
```

```
prompt> list_designs -show_file
```

In the following example, **create_design** specifies a nonunique design name:

```
prompt> create_design example
```

```
prompt> create_design example
```

SEE ALSO

- `create_bus(2)`
- `create_cell(2)`
- `create_net(2)`
- `create_port(2)`
- `current_design(2)`
- `list_designs(2)`
- `remove_bus(2)`
- `remove_cell(2)`
- `remove_design(2)`
- `remove_net(2)`
- `remove_port(2)`

create_dft_netlist

Creates a DFT netlist from the current design.

SYNTAX

```
status create_dft_netlist  
-type extest
```

ARGUMENTS

-type *extest*

Specifies the type of DFT netlist to create. There is only one valid type supported:

- **-type *extest***

This DFT netlist type removes all logic in the design except that needed to operate in the outward-facing EXTEST mode. Only the following logic types are kept: wrapper chains and wrapper control logic, interface logic between wrapper chains and I/O ports, test mode decode logic, and 1500 controller and any other logic required for EXTEST operation.

This DFT netlist type is supported only for wrapped cores.

DESCRIPTION

The **create_dft_netlist** command creates a DFT netlist from the current design. Currently only one type, **extest**, is supported

This command modifies the current design in memory. After running it, write out the DFT netlist file using the **write -format** verilog command. Because the **create_dft_netlist** command removes logic from the design in memory, these should be the last steps in your core creation script.

This command cannot be used in an ILM generation flow.

EXAMPLES

The following example creates and writes out an EXTEST-only netlist after writing the full-netlist design files:

```
# preview and insert DFT  
preview_dft -test_wrappers all
```

```
insert_dft
```

```
# write block-level files
```

```
write -hierarchy -format ddc -output block_1.ddc
```

```
write_test_model -format ddc -output block_1.ctlddc
```

```
# write an EXTEST-only netlist
```

```
create_dft_netlist -type extest
```

```
write -format verilog -hierarchy -output block_1_EXTEST.vg
```

SEE ALSO

```
set_wrapper_configuration(2)
```

```
write(2)
```

create_die_area

Creates the die area for the current design.

SYNTAX

status **create_die_area**
-coordinate *list*
-polygon *list*

ARGUMENTS

DESCRIPTION

The **create_die_area** command creates the die area for the current design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates the die area with an L-shape by defining a polygon:

```
prompt> create_die_area -polygon {{0 0} {0 400} {200 400} {200 200} \  
{400 200} {400 0} {0 0}}
```

SEE ALSO

get_die_area(2)
remove_die_area(2)

create_generated_clock

Creates a generated clock object.

SYNTAX

```
string create_generated_clock
  [-name clock_name]
  [-add]
  source_objects
  -source master_pin
  [-master_clock clock]
  [-divide_by divide_factor
   | -multiply_by multiply_factor]
  [-duty_cycle percent]
  [-invert]
  [-preinvert]
  [-edges edge_list]
  [-edge_shift edge_shift_list]
  [-combinational]
  [-comment comment_string]
```

Data Types

<i>clock_name</i>	string
<i>source_objects</i>	list
<i>master_pin</i>	list
<i>clock</i>	string
<i>divide_factor</i>	integer
<i>multiply_factor</i>	integer
<i>percent</i>	float
<i>edge_list</i>	list
<i>edge_shift_list</i>	list
<i>comment_string</i>	string

ARGUMENTS

-name *clock_name*

Specifies the name of the generated clock. If you do not use this option, the clock receives the same name as the first clock source specified in the **-source** option. If you specify the **-add** option, you must use the **-name** option and the clocks with the same source must have different names.

-add

Specifies whether to add this clock to the existing clock or to overwrite. Use this option to capture the case where multiple generated clocks must be specified on the same source, because multiple clocks fan into the master pin. Ideally, one generated clock must be specified for each clock that fans into the master pin. If you specify this option, you must also use the **-name** option.

Defining multiple clocks on the same source pin or port causes longer runtime and higher memory usage than a single clock, because the synthesis timing engine explores all possible combinations of launch and capture clocks. Use the **set_false_path** command to disable unwanted clock combinations. This option is ignored by default, unless multiple clocks analysis is enabled by setting the **timing_enable_multiple_clocks_per_reg** variable to **true**.

source_objects

Specifies a list of ports or pins defined as generated clock source objects.

-source master_pin

Specifies the master clock pin, which is either a master clock source pin or a pin in the fanout of the master clock and driving the generated clock definition pin. The clock waveform at the master pin is used for deriving the generated clock waveform.

-master_clock clock

Specifies the master clock to be used for this generated clock if multiple clocks fan into the master pin.

-divide_by divide_factor

Specifies the frequency division factor. If the *divide_factor* is 2, the generated clock period is twice as long as the master clock period.

-multiply_by multiply_factor

Specifies the frequency multiplication factor. If the *multiply_factor* is 3, the period is one third as long as the master clock period.

-duty_cycle percent

Specifies the duty cycle (in percent), if frequency multiplication is used. This is a number between 0 and 100. The duty cycle is the high pulse width.

-invert

Inverts the generated clock signal regardless of whether the sense of the source clock on the master pin is unate or non-unate (in case of frequency multiplication and division).

-preinvert

Creates a generated clock based on the inverted clock signal only when the source clock on the master pin has a non-unate sense, or the generated clock will not be inverted just as this option has not been specified. The difference between the **-invert** option and the **-preinvert** option is that the **-invert** option first creates the generated clock, then inverts the signal, and the **-preinvert** option first inverts the signal, and then creates the generated clock signal.

-edges edge_list

Specifies a list of positive integers that represents the edges from the source clock that are to form the edges of the generated clock. The edges are interpreted as alternating rising and falling edges and each edge must be not less than its previous edge. The number of edges must be an odd number and not less than 3 to make one full clock cycle of the generated clock waveform. The first edge must be greater than or equal to 1. For example, 1 represents the first source edge, 2 represents the second source edge, and so on.

-edge_shift edge_shift_list

Specifies a list of floating-point numbers that represents the amount of shift, in library time units, that the specified edges are to undergo to yield the final generated clock waveform. The number of edge shifts specified must be equal to the number of edges specified. The values can be positive or negative, with positive indicating a shift later in time, and negative a shift earlier in time.

For example, 1 indicates that the corresponding edge is to be shifted by 1 library time unit.

-combinational

Specifies that the source latency paths for this type of generated clock only includes the logic where the master clock propagates along combinational paths. The source latency paths will not flow through sequential element clock pins, transparent latch data pins, or the source pins of other generated clocks.

-comment *comment_string*

Allows the command to accept a comment string. The tool honors the annotation and preserves it with the SDC object so that the exact string is written out when the constraint is written out when you use the **write_sdc** or **write_script** command. The comment remains intact through the synthesis, place-and-route, and timing-analysis flows.

DESCRIPTION

The **create_generated_clock** command creates a generated clock object in the current design. This command defines a list of objects as generated clock sources in the current design. You can specify a pin or a port as a generated clock object. The command also specifies the clock source from which it is generated. The advantage of using this command is that whenever the master clock changes, the generated clock changes automatically.

The generated clock can be created as a frequency-divided clock with the **-divide_by** option, a frequency-multiplied clock with **-multiply_by**, or an edge-derived clock with **-edges**. In addition, the frequency-divided or frequency-multiplied clock can be inverted with the **-invert** option. The shifting of edges of the edge-derived clock is specified with the **-edge_shift** option. The **-edge_shift** option is used for intentional edge shifts and not for clock latency. If a generated clock is specified with a *divide_factor* that is a power of 2 (1, 2, 4, ...), the rising edges of the master clock are used to determine the edges of the generated clock. If the *divide_factor* is not a power of 2, the edges are scaled from the master clock edges.

Using **create_generated_clock** on an existing **generated_clock** object overwrites the attributes of the **generated_clock** object.

The **generated_clock** objects are expanded to real clocks at the time of analysis.

The following commands can reference the **generated_clock**:

```
set_clock_latency
set_clock_uncertainty
set_propagated_clock
set_clock_transition
```

To display information about generated clocks, use the **report_clock** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example creates a frequency **-divide_by** 2 generated clock:

```
prompt> create_generated_clock -divide_by 2 \
      -source CLK [get_pins test]
```

The following example creates a frequency **-divide_by 3** generated clock. If the master clock period is 30, and the master waveform is {24 36}, the generated clock period is 90 with waveform {72 108}.

```
prompt> create_generated_clock -divide_by 3 \  
-source CLK [get_pins div3/Q]
```

The following example creates a frequency **-multiply_by 2** generated clock with a duty cycle of 60%:

```
prompt> create_generated_clock -multiply_by 2 \  
-duty_cycle 60 -source CLK [get_pins test1]
```

The following example creates a frequency **-multiply_by 3** generated clock with a duty cycle equal to the master clock duty cycle. If the master clock period is 30, and the master waveform is {24 36}, the generated clock period is 10 with waveform {8 12}.

```
prompt> create_generated_clock -multiply_by 3 \  
-source CLK [get_pins div3/Q]
```

The following example creates a generated clock whose edges are edges 1, 3, and 5 of the master clock source. If the master clock period is 30, and the master waveform is {24 36}, the generated clock period is 60 with waveform {24, 54}.

```
prompt> create_generated_clock -edges {1 3 5} \  
-source CLK [get_pins test2]
```

The following example shows the generated clock in the previous example with each derived edge shifted by 1 time unit. If the master clock period is 30, and the master waveform is {24 36}, the generated clock period is 60 with waveform {25, 55}.

```
prompt> create_generated_clock -edges {1 3 5} \  
-edge_shift {1 1 1} -source CLK [get_pins test2]
```

The following example creates an inverted clock:

```
prompt> create_generated_clock -divide_by 2 -invert
```

SEE ALSO

- check_timing(2)
- create_clock(2)
- get_generated_clocks(2)
- remove_generated_clock(2)
- report_clock(2)
- set_clock_latency(2)
- set_clock_transition(2)
- set_clock_uncertainty(2)
- set_propagated_clock(2)
- timing_enable_multiple_clocks_per_reg(3)

create_link_block_abstraction

Generates IC Compiler II abstract NDM references for the current design and all the DCNXT block abstractions in the current design via link feature.

SYNTAX

```
status create_link_block_abstraction
  [-top_block]
  [-output_ddc_file output_file_name]
  [-output_ndm_dir ndm_cache_dir]
  [-overwrite]
```

Data Types

<i>output_file_name</i>	string
<i>ndm_cache_dir</i>	string

ARGUMENTS

-top_block

Creates IC Compiler II abstract for current design.

The **-top_block** option is used along with **-output_ddc_file**.

-output_ddc_file *output_file_name*

Specifies a single file into which designs are written in ddc format.

The **-top_block** option is used along with **-output_ddc_file**.

-output_ndm_dir *ndm_cache_dir*

Specifies cache directory where the IC Compiler II abstract NDMs are saved.

-overwrite

Regenerates IC Compiler II abstracts and overwrites any existing ones in the cache directory.

DESCRIPTION

This command creates IC Compiler II abstracts for the current design and all the DCNXT block abstractions present in the design

using IC Compiler II link feature. To support DCNXT block abstraction flow, the generated IC Compiler II abstracts can be used during link placer features and link reporting features.

At the top level, **set_top_implementation_options** command is used to specify the blocks that are integrated with the top-level design as block abstractions. When **create_link_block_abstraction** command is run, it creates IC Compiler II abstracts for all the blocks integrated as block abstractions. In addition to the blocks specified via **set_top_implementation_options**, the command can also create IC Compiler II abstract for current design, if **-top_block** and **-output_ddc_file** options are specified together. To create IC Compiler II abstract corresponding to current design, it's required to run **create_block_abstraction** command first to determine the interface logic. The command errors out if interface logic is not marked in the current design or the design has any unmapped and/or unplaced cells. When the IC Compiler II abstract is created for current design, the command also writes out the complete design, including the block abstraction information, into a single .ddc file specified by **-output_ddc_file** option. The generated IC Compiler II abstract will be linked to the written out .ddc file via checksum mechanism. When the current design is used as a block abstraction, user needs to provide the command written .ddc file so that the appropriate IC Compiler II abstract can be picked by the tool.

The command provides a caching mechanism so that the earlier generated IC Compiler II abstracts can be re-used. The **create_link_block_abstraction** command creates the IC Compiler II abstracts for each DCNXT block abstraction and places them in the cache. The tool generates a unique checksum for the block level .ddc files and creates a one-to-one mapping between the block level .ddc files and IC Compiler II abstracts name. If the block level .ddc files are not changing, tool can generate the IC Compiler II abstract names and pick them up from the cache directory.

The user cache directory can be specified using option **-output_ndm_dir**. If specified, the abstract NDM references will be created in that directory. If the option is not specified, the command will create a default cache directory **hier_ndm** in the current working directory and generate the IC Compiler II abstracts corresponding to each DCNXT block abstractions in the default cache directory. If the cache directory doesn't have write permission, the command will fail. The directories in **search_path** is also included in cache directories. However, these directories are read-only.

The abstracts corresponding to each block abstraction are searched in the following order of preference:

- User-specified cache directory via option **-output_ndm_dir**
- Default cache directory **hier_ndm**
- Directory list in **search_path** variable setting

If the abstract is found in multiple directories, the one based on the above order will be picked up.

When the command is triggered, it first checks if the IC Compiler II abstracts are already present in the cache directory or not. If an abstract is found for a block, no abstract generation is triggered for the same and the cached abstracts are used during link features. This helps avoiding the abstract generation every-time the command is run and helps saving runtime.

The tool builds the cache internally when the command is run. Thus, it's required to run the command even if the IC Compiler II abstracts are already present in specified cache directory, default cache directory **hier_ndm** or/and **search_Path**. If the IC Compiler II abstracts are not found in the cache directories, compile flow and report_congestion command will error out.

During link features, the tool will pick all the required IC Compiler II abstracts from the cache directories.

The IC Compiler II abstracts can be force regenerated by using **-overwrite** option. The abstracts will be written out in the default cache directory **hier_ndm**, if option **-output_ndm_dir** is not specified.

The command does not support block abstractions with transparent interface optimization.

In Milkyway mode, the command fails if **set_icc2_options** command is not run before invoking it.

The command is available only in DCNXT topographical mode.

Multicorner-Multimode Support

This command uses information from all scenarios.

EXAMPLES

The following example shows how to create and save a block abstraction:

```
prompt> create_lib -technology tech.tf -ref_libs std_ref.nlib design_nlib
prompt> set_top_implementation_options -block_references "mid_bam"
prompt> read_ddc mid_bam.ddc
prompt> read_verilog top.v
prompt> link
prompt> set_icc2_options -ref $ref_ndm_files -tech $mw_tech_file
prompt> create_link_block_abstraction
```

SEE ALSO

- check_block_abstraction(2)
- report_block_abstraction(2)
- set_top_implementation_options(2)

create_logic_net

Defines a logic net.

SYNTAX

```
status create_logic_net  
    net_name
```

Data Types

```
net_name    string
```

ARGUMENTS

net_name

Specifies the name of the net to create. You must use a simple name.

DESCRIPTION

The **create_logic_net** command creates a logic net in the active scope. It is an error if the specified net name already exists in the active scope.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

```
create_logic_port(2)  
connect_logic_net(2)
```

create_logic_port

Defines a logic port.

SYNTAX

```
string create_logic_port  
    port_name  
    [-direction in | out | inout]
```

Data Types

port_name string

ARGUMENTS

port_name

Specifies the name of the port to create. You must use a simple name.

-direction in | out | inout

Specifies the direction of the port.

The default is **in**.

DESCRIPTION

The **create_logic_port** command create a logic port on the active scope. It returns the name of the created port.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

create_logic_net(2)
connect_logic_net(2)

create_missing_constraints

Generates missing path constraints in the current design based on user-specified default settings.

SYNTAX

```
status create_missing_constraints
  [-period period_value]
  [-waveform edge_list]
  [-input_delay delay_value]
  [-output_delay delay_value]
  [-pin_load load_value]
  [-driving_lib_cell lib_cell_name]
  [-driving_lib_pin lib_pin_name]
  [-exclude object_list]
  [-clock_tracing_mode tracing_mode]
  [-output file]
  [-no_apply]
```

Data Types

```
period_value  float
edge_list     list
delay_value   float
load_value    float
lib_cell_name string
lib_pin_name  string
object_list   string
tracing_mode string
file          string
```

ARGUMENTS

-period *period_value*

Specifies the default period value for missing clocks.

-waveform *edge_list*

Specifies the rise and fall times for one default clock period.

The *period_value* and *edge_list* are passed to the **create_clock** command for generating missing clocks.

If both **-period** *period_value* and **-waveform** *edge_list* are not specified, the create missing clock feature is disabled.

-input_delay *delay_value*

Specifies default value for ports with missing input delay. If **-input_delay** *delay_value* is not specified, the create missing input delay for port feature is disabled.

The *delay_value* will be passed to the **set_input_delay** command for generating missing input delay constraints.

-output_delay *delay_value*

Specifies the default value for ports with missing output delay. If **-output_delay** *delay_value* is not specified, the create missing output delay for port feature is disabled.

The *delay_value* is passed to the **set_output_delay** command for generating missing output delay constraints.

-pin_load *load_value*

Specifies the default pin capacitance value for output ports. If **-pin_load** *load_value* is not specified, the create missing pin load for port feature is disabled.

The *load_value* value is passed to the **set_load** command for generating missing load constraints.

-driving_lib_cell *lib_cell_name*

Specifies the library cell name for ports with missing driving cells.

-driving_lib_pin *lib_pin_name*

Specifies library cell pin name for ports with missing driving cells.

The *lib_cell_name* and *lib_pin_name* names are passed to the **set_driving_cell** command for generating missing driving cell constraints.

-exclude *object_list*

Specifies the ports and clock exclusion list to skip SDC constraint generation.

-clock_tracing_mode *tracing_mode*

Specifies the tracing mode for tracing clocks from missing clock pins. The possible values are:

simple
advanced

When set to **simple** (the default value), the tool traces only ports as clocks; when set to **advanced**, the tool traces ports and register outputs as clocks.

-output *file*

Writes SDC constraints to the specified file.

-no_apply

Creates SDC constraints without applying them.

DESCRIPTION

This command generates missing path constraints in the current design based on the default settings specified with this command.

The commands *write_sdc*, *remove_sdc* and *reset_design* can be used to write out, remove, and reset SDC constraints generated by this command, respectively.

This command does nothing if no options are specified.

This command is for the DC Explorer flow only.

SEE ALSO

report_missing_constraints(2)
create_clock(2)
set_input_delay(2)
set_output_delay(2)
set_driving_cell(2)
set_load(2)
write_sdc(2)
remove_sdc(2)
reset_design(2)

create_multibit

Creates a multibit component for the specified list of cells or cell instances in the current design.

SYNTAX

```
status create_multibit  
  object_list  
  [-name multibit_name]  
  [-sort]  
  [-no_sort]
```

Data Types

```
object_list    list  
multibit_name  string
```

ARGUMENTS

object_list

Specifies a list of cells in the current design for which a multibit component is to be created.

-name *multibit_name*

Specifies the name to be given to the new multibit component. If this option is omitted, the name of the first cell in the component is used to generate a name for the multibit component. If a multibit component by the name *multibit_name* already exists in the current design, an error is issued and the command terminates.

-sort

Sorts the cells specified in *object_list* in alphanumeric order. By default, cells are sorted in reverse alphanumeric order of their names.

-no_sort

Specifies that the order of the cells in the multibit component should be the same as specified on the command line. By default, cells are sorted in reverse alphanumeric order of their names.

DESCRIPTION

The **create_multibit** command creates a new multibit component and each cell or instance in *object_list* is inserted into that multibit component. If a name is provided, the new multibit component gets that name; otherwise, a name is generated internally. If a cell

instance is given, the parent of the cell instances should be the same and the parent must be unique.

If the cells in the list are MUX_OP cells, a separate multibit component is created for each cell.

The list of cells is sorted in a reverse alphanumeric order by default. Use the **-sort** option to sort in alphanumeric order, or the **-no_sort** order to order cells in the same way as specified on the command line.

This command requires the design to be linked. If an attempt to link the design fails, the command terminates. If any of the cells in *object_list* already belong to a multibit component, an error is issued and the command terminates.

EXAMPLES

In the following example, a multibit component is created for four registers:

```
prompt> report_multibit
```

```
*****
Report : multibit
Design : subtest
Version: 1998.02
Date   : Fri Aug 29 10:01:19 1997
*****
```

Attributes:

```
b - black box (unknown)
h - hierarchical
n - noncombinational
r - removable
u - contains unmapped logic
```

Total 0 Multibit Components

```
prompt> create_multibit -name y_reg {y_reg*}
1
```

```
prompt> report_multibit
```

```
*****
Report : multibit
Design : subtest
Version: 1998.02
Date   : Fri Aug 29 10:01:19 1997
*****
```

Attributes:

```
b - black box (unknown)
h - hierarchical
n - noncombinational
r - removable
u - contains unmapped logic
```

Multibit Component : y_reg

Cell	Reference	Library	Area	Width	Attributes

y_reg[3]	**SEQGEN**	0.00	1	n, u
y_reg[2]	**SEQGEN**	0.00	1	n, u
y_reg[1]	**SEQGEN**	0.00	1	n, u
y_reg[0]	**SEQGEN**	0.00	1	n, u

Total 4 cells		0.00	4	

Total 1 Multibit Components

In the next example, the command is run on a list of registers and uses the **-no_sort** option:

```
prompt> create_multibit -name y_reg \
      -no_sort {y_reg[0] y_reg[2] y_reg[1] y_reg[3]}
1
```

```
prompt> report_multibit
```

```
*****
Report : multibit
Design : subtest
Version: 1998.02
Date   : Fri Aug 29 10:01:19 1997
*****
```

Attributes:

- b - black box (unknown)
- h - hierarchical
- n - noncombinational
- r - removable
- s - synthetic operator
- u - contains unmapped logic

Multibit Component : y_reg

Cell	Reference	Library	Area	Width	Attributes
y_reg[0]	**SEQGEN**		0.00	1	n, u
y_reg[2]	**SEQGEN**		0.00	1	n, u
y_reg[1]	**SEQGEN**		0.00	1	n, u
y_reg[3]	**SEQGEN**		0.00	1	n, u

Total 4 cells			0.00	4	

Total 1 Multibit Components

In the next example, the command is run on a list of MUX_OP cells:

```
prompt> create_multibit {U4, U7}
1
```

```
prompt> report_multibit
```

```
*****
Report : multibit
Design : subtest
Version: 1998.02
Date   : Fri Aug 29 10:01:19 1997
*****
```

Attributes:

b - black box (unknown)
 h - hierarchical
 n - noncombinational
 r - removable
 s - synthetic operator
 u - contains unmapped logic

Multibit Component : y_reg

Cell	Reference	Library	Area	Width	Attributes
U4	*MUX_OP_4_2.2		0.00	2	s, u
Total 1 cells			0.00	4	

Multibit Component : U7_multibit

Cell	Reference	Library	Area	Width	Attributes
U7	*MUX_OP_4_2.2		0.00	2	s, u
Total 1 cells			0.00	2	

Total 2 Multibit Components

In the next example, the command is run using cell instances U1/test1 and U1/test2. In this case the instance U1 corresponding to design MID is unique.

```
prompt> create_multibit {U1/test1 U1/test2} -name mult
1
```

```
prompt> report_multibit
```

```

*****
Report : multibit
Design : subtest
Version: 1998.02
Date   : Fri Aug 29 10:01:19 1997
*****
```

Attributes:

b - black box (unknown)
 h - hierarchical
 n - noncombinational
 r - removable
 s - synthetic operator
 u - contains unmapped logic

Multibit Component : mult

Cell	Reference	Library	Area	Width	Attributes
U1/test1	**SEQGEN**		0.00	1	n, u
U1/test2	**SEQGEN**		0.00	1	n, u
Total 2 cells			0.00	2	

Total 1 Multibit Components

SEE ALSO

- `current_design(2)`
- `remove_multibit(2)`
- `report_cell(2)`
- `report_compile_options(2)`
- `report_multibit(2)`
- `report_reference(2)`

create_mw_lib

Creates a Milkyway library.

SYNTAX

```
status create_mw_lib
  [-technology technology_file_name]
  [-bus_naming_style style]
  [-mw_reference_library lib_list]
  [-reference_control_file rc_file_name]
  [-open]
  libName
```

Data Types

<i>technology_file_name</i>	string
<i>style</i>	string
<i>lib_list</i>	string
<i>rc_file_name</i>	string
<i>libName</i>	string

ARGUMENTS

-technology *technology_file_name*

Specifies the name of the technology file for the newly created Milkyway library. This option and **-plib** are mutually exclusive.

-bus_naming_style *style*

Specifies the bus naming style for the library. The default bus naming style is [%d].

-mw_reference_library *lib_list*

Specifies a list of reference libraries to be used for the new library.

-reference_control_file *rc_file_name*

Specifies the reference control file used to set reference library information for the new library.

-open

Opens the library after creation.

libName

Specifies the name of the Milkyway library to be created.

DESCRIPTION

This command creates a Milkyway library.

The **-technology** and **-plib** options are mutually exclusive, and at least one of them must be specified.

Some advanced technologies require more than the default 255 layers. In these cases, you can extend the number of layers to 4095 by executing the **extend_mw_layers** command before you execute the **create_mw_lib** command. Once created in the default 255-layer mode or extended 4095-layer mode, the Milkyway library must remain in that layer mode. For details, see the man page for the **extend_mw_layers** command.

By default, the newly created Milkyway library is not open in the current session. To manipulate it, first run the **open_mw_lib** command. The library can also be opened by using the **-open** option, but to make the scripts more reusable, use **open_mw_lib**.

All strings stored in this library, as well as in designs belonging to it, are case-sensitive, and all string operations are performed as case-sensitive.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a Milkyway design library with a technology file and bus naming style as [%d].

```
prompt> create_mw_lib design -technology test.tf \  
        -bus_naming_style {[%d]}  
1
```

SEE ALSO

close_mw_lib(2)
extend_mw_layers(2)
open_mw_lib(2)

create_net

Creates nets in the current design or its subdesign.

SYNTAX

```
status create_net  
  [-power |  
   -ground]  
  net_list
```

Data Types

net_list list

ARGUMENTS

-power

Creates a power net. By default, the tool creates a signal net. This option is available only in Design Compiler topographical mode.

The **-power** and **-ground** options are mutually exclusive; you can specify only one.

-ground

Creates a ground net. By default, the tool creates a signal net. This option is available only in Design Compiler topographical mode.

The **-power** and **-ground** options are mutually exclusive; you can specify only one.

net_list

Specifies the names of the created nets. If you specify a hierarchical net name, the net is created in the specified instance. The net name must be unique within the current design or the subdesign where it is created. If you use a hierarchical net name, the parent instance must be unique; it cannot be an instance of a multiply-instantiated design.

This option is required.

DESCRIPTION

The **create_net** command creates new net objects in the current design or its subdesign based on the *net_list* argument. The **create_net** command creates only scalar (single bit) nets.

To bundle scalar nets into buses, use the **create_bus** command.

Nets connect pins and ports in a design. When you create nets with **create_net**, they are not connected. To establish this connection, use the **connect_net** command. To remove nets from the current design, use the **remove_net** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **create_net** to create net objects in the current design:

```
prompt> current_design
```

```
prompt> create_net {N1 N2 N3 N4}
```

```
prompt> get_nets *
```

The following example uses **create_net** to create net objects in the mid subdesign. Note that mid1 is an instance of the mid design. The mid design must be unique for the **create_net** command to succeed.

```
prompt> current_design
```

```
prompt> create_net {mid1/N1 mid1/N2 mid1/N3 mid1/N4}
```

```
prompt> get_nets mid1*
```

SEE ALSO

- all_connected(2)
- connect_net(2)
- current_design(2)
- disconnect_net(2)
- remove_bus(2)
- remove_net(2)

create_net_search_pattern

Defines a net pattern.

SYNTAX

```
pattern_id create_net_search_pattern
[-fanout_upper_limit number]
[-fanout_lower_limit number]
[-bbox_half_perimeter_upper_limit length]
[-bbox_half_perimeter_lower_limit length]
[-net_length_upper_limit length]
[-net_length_lower_limit length]
[-blocked_area_ratio_upper_limit percentage]
[-blocked_area_ratio_lower_limit percentage]
[-aspect_ratio_upper_limit ratio]
[-aspect_ratio_lower_limit ratio]
[-centered_within {llx lly urx ury}]
[-connect_to_port]
[-connect_to_macro]
[-setup_slack_lower_limit number]
[-setup_slack_upper_limit number]
```

Data Types

<i>number</i>	integer
<i>length</i>	float
<i>percentage</i>	float
<i>ratio</i>	float
<i>llx</i>	float
<i>lly</i>	float
<i>urx</i>	float
<i>ury</i>	float

ARGUMENTS

-fanout_upper_limit *number*

Identifies nets with fanouts that are less than the specified value.

-fanout_lower_limit *number*

Identifies nets with fanouts that are greater than or equal to the specified value.

-bbox_half_perimeter_upper_limit *length*

Identifies nets whose bounding box width and height have a sum that is less than the specified value. A net's bounding box is the smallest rectangle that contains all of its pins. The measurement unit is microns.

-bbox_half_perimeter_lower_limit length

Identifies nets whose bounding box width and height have a sum that is greater than or equal to the specified value. A net's bounding box is the smallest rectangle that contains all of its pins. The measurement unit is microns.

-net_length_upper_limit length

Identifies nets that have a total length that is less than the specified value. If a net is only virtually routed, the sum of its `x_length` and `y_length` attributes is checked; however, if a net is detail routed, its `route_length` attribute is checked. The measurement unit is microns.

-net_length_lower_limit length

Identifies nets that have a total length that is greater than or equal to the specified value. If a net is only virtually routed, the sum of its `x_length` and `y_length` attributes is checked; however, if a net is detail routed, its `route_length` attribute is checked. The measurement unit is microns.

-blocked_area_ratio_upper_limit percentage

Identifies nets whose bounding boxes are covered with blockages by a fractional amount that is less than the specified value. For example, if you want to create a collection of nets that have less than 1% of their bounding box area covered by some kind of blockage polygon (such as hard or soft placement blockages), use **-blocked_area_ratio_upper_limit 0.01**. Note that cells returned by the **all_macro_cells** command are considered sources of blockage. Also, partial placement blockages are NOT considered for this calculation. A net's bounding box is the smallest rectangle that contains all of its pins.

-blocked_area_ratio_lower_limit percentage

Identifies nets whose bounding boxes are covered with blockages by a fractional amount that is greater than or equal to the specified value. For example, if you want to create a collection of nets that have at least 50% of their bounding box area covered by some kind of blockage polygon (such as hard or soft placement blockages), then specify **-blocked_area_ratio_lower_limit 0.5**. Note that cells returned by the **all_macro_cells** command are considered sources of blockage. Also, partial placement blockages are NOT considered for this calculation. A net's bounding box is the smallest rectangle that contains all of its pins.

-aspect_ratio_upper_limit ratio

Identifies nets whose bounding boxes have aspect ratios that are less than the specified value. A net's aspect ratio is its bounding box (horizontal) width divided by its (vertical) height. A net's bounding box is the smallest rectangle that contains all of its pins. If a bounding box's width or height is less than 0.05, it is replaced with 0.05 for this specific calculation.

-aspect_ratio_lower_limit ratio

Identifies nets whose bounding boxes have aspect ratios that are greater than or equal to the specified value. A net's aspect ratio is its bounding box (horizontal) width divided by its (vertical) height. A net's bounding box is the smallest rectangle that contains all of its pins. If a bounding box's width or height is less than 0.05, it is replaced with 0.05 for this specific calculation.

-centered_within {llx lly urx ury}

Identifies nets whose bounding boxes have a center x and center y that fall within the specified region. A net's bounding box is the smallest rectangle that contains all of its pins. The coordinate values must be in microns.

-connect_to_port

Identifies all nets that are connected to a design's primary I/Os (all the ports of the top module of the netlist). If this option is specified with **-connect_to_macro**, a collection of all nets connected to BOTH primary and macro I/Os is derived.

-connect_to_macro

Identifies all nets that are connected to the I/Os of a design's macro pins. If this option is specified with **-connect_to_port**, a

collection of all nets connected to BOTH primary and macro I/Os is derived.

-setup_slack_lower_limit *number*

Identifies nets in the specified pattern ID that have a setup slack that is greater than or equal to the specified value. A net's setup slack is the worst setup slack seen across all its sinks. To select all of the nets in pattern 1 that have a setup slack of more than 100, use **get_matching_nets_for_pattern -setup_slack_lower_limit 100 -pattern 1**.

-setup_slack_upper_limit *number*

Identifies nets in the specified pattern ID that have a setup slack that is less than the specified value. A net's setup slack is the worst setup slack seen across all its sinks. To select all of the nets in pattern 2 that have a setup slack of less than 0, use **get_matching_nets_for_pattern -setup_slack_upper_limit 0 -pattern 2**.

DESCRIPTION

The **create_net_search_pattern** command defines a net pattern, which is a matching constraint that simultaneously uses several different physical attributes.

After they are created, net patterns can be used in the following ways:

- Net identification, to establish collections of nets.
- Specifies minimum and maximum layers when you run **compile_ultra -spg -layer_optimization** to alter the unit resistance and capacitance of the matched nets in the pattern-based collections.

Other commands can use patterns, such as **set_net_search_pattern_delay_estimation_options** and **get_matching_nets_for_pattern**, both of which have the **-pattern** option.

Whenever a pattern is created, it is assigned an integer ID. If the command fails, a -1 is returned. If the command succeeds, the assigned pattern ID is returned. The ID must be referenced for subsequent operations. For example, if you create a pattern of all nets with lengths between 5 and 10 um (for example **create_net_search_pattern -fanout_lower_limit 5 -fanout_upper_limit 10**) and it is assigned a pattern ID of 1, you can then enter **get_matching_nets_for_pattern -pattern 1** to collect nets that match the pattern.

Note that pattern IDs always start at 1 and can never be reused even if a pattern ID is removed. Each time a pattern is created, the pattern ID count always increases by 1. Although there is no hard-coded upper limit, it is strongly recommended to keep the pattern count under 5,000. Note that all net search pattern-related data is saved in the Milkyway database for future reference.

Multicorner-Multimode Support

This command uses information from all active scenarios. For setup slack, the worst value from all active scenarios is always used.

EXAMPLES

In the following example, the **create_net_search_pattern** command creates a collection of all nets that have a length of 200 um or more and also touch primary I/O pins:

```
prompt> create_net_search_pattern -net_length_lower_limit 200
      -connect_to_port
```

SEE ALSO

report_net_search_pattern(2)
remove_net_search_pattern(2)
set_net_search_pattern_delay_estimation_options(2)
report_net_search_pattern_delay_estimation_options(2)
set_net_search_pattern_priority(2)
report_net_search_pattern_priority(2)
get_matching_nets_for_pattern(2)

create_net_shape

Creates a new net shape.

SYNTAX

```
status create_net_shape
  [-type wire | path | rect]
  -origin point
  | -bbox rect
  | -points list_of_points
  [-length length]
  [-width width]
  [-path_type square | round | extend_half_width | octagon]
  -layer layer
  [-mask_constraint constraint]
  -net net_name
  [-vertical]
  [-route_type route_type]
  [-net_type ground | power | clock | signal]
  [-datatype int]
  [-avoid_short_segment]
```

Data Types

<i>point</i>	point
<i>rect</i>	rectangle
<i>list_of_points</i>	list of points
<i>length</i>	float
<i>width</i>	float
<i>layer</i>	collection
<i>constraint</i>	string
<i>net_name</i>	string
<i>route_type</i>	string
<i>int</i>	integer

ARGUMENTS

-type wire | path | rect

Specifies the type of the net shape. Valid values are **wire**, **path**, and **rect** (rectangle).

If you do not specify this option, the tool determines the type by using the following rules, in order of precedence:

1. If you use the **-origin** option, the net shape is **wire**.

The wire is horizontal, unless you also specify the **-vertical** option.

2. If you use the **-bbox** option, the net shape is **wire** if you also use the **-path_type**, **-route_type**, or **-vertical** options. If you do not use any of these additional options, the net shape is **rect**.
3. If you use the **-points** option, the net shape is **path**.

-origin *point*

Specifies the origin of the net shape for wires.

When you specify this option, you must also specify the **-length** and **-width** options.

The **-origin**, **-points**, and **-bbox** options are mutually exclusive. You must specify one of these options.

-bbox *rect*

Specifies the bounding box of the net shape.

You must specify the lower-left and upper-right corners of the rectangle by using the following syntax: `{{llx lly} {urx ury}}`.

The **-origin**, **-points**, and **-bbox** options are mutually exclusive. You must specify one of these options.

-points *list_of_points*

Specifies the point sequence of the net shape for paths.

When you specify this option, you must also specify the **-width** option.

The **-origin**, **-points**, and **-bbox** options are mutually exclusive. You must specify one of these options.

-length *length*

Specifies the length of the net shape for wires in user units.

This option is required when you specify the **-origin** option and ignored otherwise.

-width *width*

Specifies the width of the net shape for wires in user units.

This option is required when you specify the **-origin** or **-points** option and ignored otherwise.

-path_type *square* | *round* | *extend_half_width* | *octagon*

Specifies the alignment type of a wire or path end.

The default is **square**.

You can specify one of the following values:

- **square** (square, no extension)
- **round** (round, half-width extension)
- **extend_half_width** (square, half-width extension)
- **octagon** (octagon, half-width extension)

This option can be used only for wire or path shapes.

-layer *layer*

Specifies the layer for the net shape. You can specify the layer by using the layer name from the technology file.

This option is required.

-mask_constraint *constraint*

Specifies the mask constraint for the net shape.

The valid mask constraints are **any_mask**, **mask1_soft**, **mask1_hard**, **mask2_soft**, **mask2_hard**, **same_mask**, **mask3_soft**, and **mask3_hard**.

-net *net_name*

Specifies the net associated with the net shape.

This option is required.

-vertical

Indicates that the net shape is vertical.

By default, the net shape is horizontal.

-route_type *route_type*

Specifies the route type of the net shape.

By default, the route type is **signal_route**.

You can specify one of the following values:

- **user_enter** (user entered)
- **signal_route** or **signal_route_detail** (detail routing)
- **signal_route_global** (global routing)
- **pg_ring** (power or ground ring)
- **pg_strap** (power or ground strap)
- **pg_macro_io_pin_conn** (power or ground net that connects to the power or ground pin of an I/O pad cell or a macro cell)
- **pg_std_cell_pin_conn** (power or ground net that connects to the power or ground pin of a standard cell)
- **clk_ring** (clock ring)
- **clk_strap** (clock strap)
- **clk_zero_skew_route** (clock zero-skew route)
- **bus** (bus)
- **shield** or **shield_fixed** (fixed shield)
- **shield_dynamic** (dynamic shield)
- **fill_track** or **clk_fill_track** (fill track)

This option can be used only for wire, path, or rectangle shapes.

-net_type *ground | power | clock | signal*

Specifies the type of the net on which to create the net shape. The valid net types are **ground**, **power**, **clock**, or **signal**.

-datatype *int*

Specifies the data type number in the range 0-255.

By default, the data type is 0.

-avoid_short_segment

Avoids creating segments shorter than a half width for paths.

DESCRIPTION

This command creates a shape object that is attached to the specified net. It returns 1 if success or 0 if command fails. Shapes will be created during compile.

The valid shape types are wire (horizontal or vertical), path, and rectangle.

Note: For wires, the shape must be symmetrical about the center line. If you specify the width as an odd value, either explicitly by using the **-width** option or implicitly by using the **-bbox** option, the tool rounds the width down to an even value.

For example, the following command specifies a width of 9995 database units. If the technology file defines the lengthPrecision as 1000, the tool rounds this down to 9994, resulting in a bounding box with the coordinates (100.000, 100.000) (**109.994**, 169.895).

```
prompt> create_net_shape -net VDD -type wire -route_type pg_strap \
-layer M6 -bbox { {100.000 100.000} {109.995 169.895} } -vertical
```

If the **lengthPrecision** attribute is set to 1000 in the technology file, the width in database units is 9995. The tool rounds this down to 9994, which results in a bounding box of { {100.000 100.000} {**109.994** 169.895} }.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a wire net shape:

```
prompt> create_net_shape -type wire -net VSS \
-origin {0 0} -length 10 -width 2 -layer M1
```

The following example creates a path net shape:

```
prompt> create_net_shape -type path -net {VSS} \
-path_type extend_half_width -layer M3 \
-points {{845 715} {845 710} {860 710}} -width 0.230
```

The following example creates a rectangle net shape:

```
prompt> create_net_shape -type rect -net {CLOCK_Net27} \
-bbox {{76 110} {82 115}} -layer METAL1
```

SEE ALSO

`create_placement_blockage(2)`
`create_route_guide(2)`
`create_terminal(2)`
`create_user_shape(2)`
`create_via(2)`

create_operating_conditions

Creates a new set of operating conditions in a library.

SYNTAX

```
status create_operating_conditions
-name name
-library library_name
-process process_value
-temperature temperature_value
-voltage voltage_value
[-tree_type tree_type]
[-calc_mode calc_mode]
[-rail_voltages rail_value_pairs]
```

Data Types

<i>name</i>	string
<i>library_name</i>	string
<i>process_value</i>	float
<i>temperature_value</i>	float
<i>voltage_value</i>	float
<i>tree_type</i>	string
<i>calc_mode</i>	string
<i>rail_value_pairs</i>	list

ARGUMENTS

-name *name*

Specifies the name of the new set of operating conditions.

-library *library_name*

Specifies the name of the library for the new operating conditions.

-process *process_value*

Specifies the process scaling factor for the operating conditions. Allowed values are 0.0 through 100.0.

-temperature *temperature_value*

Specifies the temperature value, in degrees Celsius, for the operating conditions. Allowed values are -300.0 through +500.0.

-voltage *voltage_value*

Specifies the voltage value for the operating conditions. Allowed values are 0.0 through 1000.0.

-tree_type *tree_type*

Specifies the tree type for the operating conditions. Allowed values are **balanced_tree** (the default), **best_case_tree**, or **worst_case_tree**. The tree type is used to estimate interconnect delays by providing a model of the RC tree.

-calc_mode *calc_mode*

For use only with DPCM libraries. Specifies the DPCM delay calculator mode for the operating conditions. Allowed values are **unknown** (the default), **best_case**, **nominal**, or **worst_case**. If you use the default value, the **worst_case** value will be used during analysis. If **-rail_voltages** is specified, the command sets corresponding (**worst_case**, **nominal**, and **best_case**) voltage values.

-rail_voltages *rail_value_pairs*

Specifies a list of name-value pairs that defines the voltage for each specified rail. The name is one of the rail names defined in the library; the value is the voltage to be assigned to that rail. By default, rail voltages are determined by the values given to them in the library. Use this option to override the default voltages for the specified rails.

DESCRIPTION

The **create_operating_conditions** command creates a new set of operating conditions in the specified library. A technology library contains a fixed set of operating conditions. This command allows you to create new and additional operating conditions.

To see the operating conditions defined for a library, use the **report_lib** command.

To set operating conditions on the current design, use the **set_operating_conditions** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a new set of operating conditions called *WC_CUSTOM* in the library named *tech_lib*, specifying new values for process, temperature, voltage, parameter1, and tree type. By default, rail voltages remain as defined in the library.

```
prompt> create_operating_conditions -name WC_CUSTOM \
      -library tech_lib -process 1.2 -temperature 30.0 -voltage 2.8 \
      -parameter1 1.2 -tree_type worst_case_tree
```

The following example creates a new set of operating conditions called *OC3* in the library named *IBM_CMOS5S6_SC*, specifying new values for process, temperature, voltage, and rail voltages for rails VTT and VDDQ. By default, the tree type **balanced_tree** is used.

```
prompt> create_operating_conditions -name OC3 \
      -lib IBM_CMOS5S6_SC -proc 1.0 -temp 100.0 -volt 4.0 \
      -rail_voltages {VTT 3.5 VDDQ 3.5}
```


SEE ALSO

report_lib(2)
set_operating_conditions(2)

create_pass_directories

Creates the directory structure required for storing Automated Chip Synthesis data.

SYNTAX

```
int create_pass_directories
    pass_list
```

Data Types

```
pass_list  list
```

ARGUMENTS

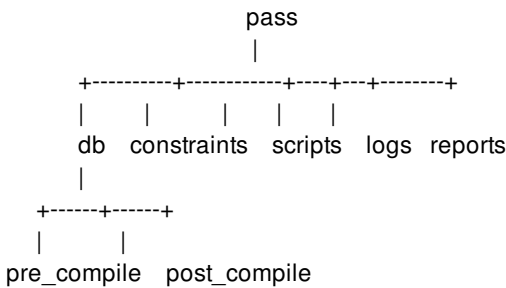
pass_list

Specifies a list of directory names that specify the passes that require data directories.

DESCRIPTION

The **create_pass_directories** command creates a directory structure to store pass data for each specified pass. The pass data directories are subdirectories to the Automated Chip Synthesis working directory, specified by the **acs_work_dir** variable.

The directory structure for each pass data directory is:



If any part of the data directory structure for a specified pass does not exist, Automated Chip Synthesis creates it.

The @ character is reserved to specify pass-dependent directory names. It cannot be used as a literal character in Automated Chip Synthesis directory paths.

SEE ALSO

`remove_pass_directories(2)`

`acs_work_dir(3)`

create_pin_guide

Creates a pin guide for top-level ports to constrain the terminals to a specified bounding box.

SYNTAX

```
collection create_pin_guide
  -bbox bounding_box_rect
  | -boundary rectilinear_boundary
  [-parents soft_macro_or_plan_group]
  [-name pin_guide_name]
  [-exclusive]
  objects
```

Data Types

<i>bounding_box_rect</i>	list of points
<i>rectilinear_boundary</i>	list of points
<i>soft_macro_or_plan_group</i>	collection
<i>pin_guide_name</i>	string
<i>objects</i>	collection

ARGUMENTS

-bbox *bounding_box_rect*

Specifies the bounding box of a rectangular pin guide. The format of the bounding box specification is $\{\{lx\ ly\} \{urx\ ury\}\}$, which specifies the lower-left and upper-right corners of the rectangle. The unit is in microns.

The **-boundary** and **-bbox** options are mutually exclusive. You must specify one of these options.

-boundary *rectilinear_boundary*

This option is not supported in Design Compiler in topographical mode and is ignored if specified.

-parents *soft_macro_or_plan_group*

This option is not supported in Design Compiler in topographical mode and is ignored if specified.

-name *pin_guide_name*

Specifies the name of the pin guide to be created.

If this option is not used, a default name is given to the pin guide.

-exclusive

Prevents the placement of terminals or soft macro pins within the pin guide, except for those that are specified to be placed inside the pin guide at the time it is created.

If nets are specified at the time of pin guide creation, the only pins allowed inside the pin guide are those that are connected to the nets or to the feedthrough pins derived from the global routes of these nets.

objects

Specifies the port objects to be placed into the pin guides.

DESCRIPTION

This command creates a pin guide for a plan group or soft macro to constrain the terminals generated by pin assignment to a specified region. The command returns a collection that contains the created pin guide.

Pin guide shapes can be used to define feedthroughs, that is, added ports that do not correspond to existing logical ports, to route a net through a plan group or soft macro. When used in this way, the collection of objects supplied to the command should consist only of nets.

If a parent block of the pin guide contains a connection to the net and the pin guide crosses the block boundary more than once, a port is associated with each crossing. One of the ports corresponds to the original port. Any additional ports created are feedthrough ports.

If a parent block of the pin guide does not contain any connection to the net, the pin guide must cross the block boundary in at least two separate locations. The net is forced to route through this block and has a feedthrough port in each of the areas where the guide crosses the block boundary. If a pin guide only crosses an unconnected block in one place, it is considered an error and this portion of the pin guide is ignored. The pin guide should be a single contiguous shape (not disjoint), and can be either rectangular or rectilinear.

You should ensure that the pin guide represents a reasonable path for the feedthrough routing. For any pin guide, the number of crossings of the block boundary must be less than or equal to the number of net targets outside the block. For example, if the pin guide is an irregular shape that crosses the block boundary in three disjoint areas, there must be at least three separate targets (ports) on the net outside the block, so that each new feedthrough port can be connected to at least one external target. If there are fewer external targets, say two in this example, it is considered an error and this portion of the pin guide is ignored.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a pin guide named abc for all ports whose name contains the string "clk" on the plan group named mem1.

```
prompt> set pin_guide \
[create_pin_guide -bbox {0 0 100 100} -name abc \
[get_ports *clk*]]
```

SEE ALSO

create_placement_blockage

Creates a new placement blockage.

SYNTAX

```
status create_placement_blockage
  -bbox {llx1 lly1 urx1 ury1}
  [-type hard | soft | partial]
  [-blocked_percentage percentage]
  [-no_register]
  [-buffer_only]
  [-category attribute_name]
  [-no_rp_group]
  [-no_pin]
  [-blocked_layers layers]
  [-no_hard_macro]
  [-name blockage_name]
```

Data Types

<i>llx1</i>	float
<i>lly1</i>	float
<i>urx1</i>	float
<i>ury1</i>	float
<i>percentage</i>	integer
<i>layers</i>	string
<i>blockage_name</i>	string

ARGUMENTS

-bbox {*llx1 lly1 urx1 ury1*}

Specifies the coordinates of the bounding box of the blockage.

-type hard | soft | partial

Specifies the type of blockage to be created.

Valid values are:

- **hard**
With a hard blockage, the placer does not place any standard cells or hard macros in the specified region. This is the default value.
- **soft**

With a soft blockage, the coarse placer does not place any standard cells or hard macros in the specified region. However, cells might be placed there during optimization or legalization.

- **partial**

With a partial blockage, the amount of area used to place cells is limited to the specified percent of the blockage area.

Note that when you create a partial blockage, you must also specify the blocked percentage value by using the **-blocked_percentage** option.

This option is mutually exclusive with the **-no_hard_macro** and **-no_pin** options.

-blocked_percentage *percentage*

Specifies the percentage blockage for a partial blockage. To allow cell density of 100 percent in the specified partial blockage area, specify the **-blocked_percentage 0** option.

This option can only be used with the **partial** blockage type.

-no_register

Creates a register blockage, which prevents the coarse placer from placing any register cells within the specified blockage area.

The **-no_register** option must be used with the **-type partial** option. The coarse placer honors the attribute set by this option.

This option is mutually exclusive with the **-type hard**, **-type soft**, **-no_hard_macro**, **-no_rp_group**, **-buffer_only**, **-category**, and **-no_pin** options.

-buffer_only

Creates a buffer-only blockage, which prevents the coarse placer from placing any non-buffer cells within the specified blockage area.

The **-buffer_only** option must be used with the **-type partial** option. The coarse placer honors the attribute set by this option. In addition, the **placer_enable_redefined_blockage_behavior** variable must have a value of **true**, which is the default value.

This option is mutually exclusive with the **-type hard**, **-type soft**, **-no_hard_macro**, **-no_rp_group**, **-no_register**, **-category**, and **-no_pin** options.

-category *attribute_name*

Creates a category blockage, which prevents the coarse placer from placing a user-specified category of cells within the specified blockage area.

You specify the category for a cell by creating a Boolean user attribute named *attribute_name* on instances or references. If the attribute is true for an instance or its reference, the instance is prohibited within the blockage area. If the attribute is present on both the instance and the reference, the tool uses the instance value.

The **-category** option must be used with the **-type partial** option. The coarse placer honors the attribute set by this option. In addition, the **placer_enable_redefined_blockage_behavior** variable must have a value of **true**, which is the default value.

This option is mutually exclusive with the **-type hard**, **-type soft**, **-no_hard_macro**, **-no_rp_group**, **-no_register**, **-buffer_only**, and **-no_pin** options.

-no_rp_group

Creates a relative placement group blockage, which prevents the placement of relative placement groups within the specified area. This option affects only cells that belong to relative placement groups.

The **-no_rp_group** option must be used with the **-type partial** option.

This option is mutually exclusive with the **-type hard**, **-type soft**, **-no_hard_macro**, **-no_register**, **-buffer_only**, **-category**, and **-no_pin** options.

-no_pin

Specifies the pin blockage area where the global router does not route and the pin placer does not assign pins during pin assignment.

Note that when you create a pin blockage, you can also specify the blocked layers by using the **-blocked_layers** option. If you do not specify this option, the default is to block all layers.

This option is mutually exclusive with the **-no_hard_macro**, **-no_rp_group** **-no_register**, **-buffer_only**, **-category**, and **-type** options.

The **-no_pin** option can be read, written and reported in Design Compiler Topographical mode, but it does not impact placement and floorplan estimation in Design Compiler Topographical mode.

-blocked_layers *layers*

Specifies the layers for which routing to pins is blocked.

This option can only be used with the **-no_pin** option.

The **-blocked_layers** option can be read, written and reported in Design Compiler Topographical mode, but it does not impact placement and floorplan estimation in Design Compiler Topographical mode.

-no_hard_macro

Specifies the hard macro blockage area where the placer does not place hard macros.

This option is mutually exclusive with the **-no_pin**, **-no_rp_group** **-no_register**, **-buffer_only**, **-category**, and **-type** options.

The **-no_hard_macro** option can be read, written and reported in Design Compiler Topographical mode, but it does not impact placement and floorplan estimation in Design Compiler Topographical mode.

-name *blockage_name*

Specifies the optional name of the blockage. If you specify a name for the blockage, you can get the blockage by the name later in the flow.

DESCRIPTION

This command creates a new placement blockage that is used to control the placement of cells in a specified rectangular area.

Snapping to the bounding box is done automatically using global snap settings.

See the description of the **-type** option and the relevant placer documentation for more information.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a soft placement blockage.

```
prompt> create_placement_blockage -bbox {0 0 100 100} -type soft
```

The following example creates a category blockage that prohibits instance U1 and any instances that reference my_lib/sdffx1 within the specified region, but allows instance U2.

```
prompt> define_user_attribute -type boolean -class cell my_attr  
prompt> define_user_attribute -type boolean -class lib_cell my_attr  
prompt> set_attribute U1 my_attr true  
prompt> set_attribute U2 my_attr false  
prompt> set_attribute my_lib/sdffx1 my_attr true  
prompt> create_placement_blockage -type partial -category my_attr  
      -blocked_percentage 30 -bbox {0 0 100 100} -name my_category_blockage
```

SEE ALSO

- `create_net_shape(2)`
- `create_route_guide(2)`
- `create_terminal(2)`
- `create_user_shape(2)`
- `create_via(2)`
- `placer_enable_redefined_blockage_behavior(3)`

create_port

Creates ports in the current design or its subdesign.

SYNTAX

```
status create_port  
  port_list  
  [-direction dir]
```

Data Types

```
port_list  list  
dir        string
```

ARGUMENTS

port_list

Specifies names of ports created in the current design. Each port name must be unique within the current design.

-direction *dir*

Specifies the signal flow of the created port. The possible values are **in**, **out**, or **inout**.

The default is **in**.

DESCRIPTION

The **create_port** command creates new port objects in the current design or its subdesign. The **create_port** command creates only scalar or single bit ports.

To bundle scalar ports into buses, use the **create_bus** command.

Ports are the external connection points on a design. To connect ports to nets inside a design, use the **connect_net** command. To remove ports from the current design, use the **remove_port** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **create_port** to create ports in the current design:

```
prompt> current_design
Current design is 'TOP'.
{TOP}

prompt> create_port -direction "in" {A1 A2 A3 A4}
Creating port 'A1' in design 'TOP'.
Creating port 'A2' in design 'TOP'.
Creating port 'A3' in design 'TOP'.
Creating port 'A4' in design 'TOP'.
1

prompt> get_ports *
{A1 A2 A3 A4}
```

The following example uses **create_port** to create ports in the *MID* subdesign. *U1* is a unique instance of the design named *MID*.

```
prompt> current_design
Current design is 'TOP'.
{TOP}

prompt> create_port -direction "in" {U1/A1 U1/A2 U1/A3 U1/A4}
Creating port 'A1' in design 'MID'.
Creating port 'A2' in design 'MID'.
Creating port 'A3' in design 'MID'.
Creating port 'A4' in design 'MID'.
1

prompt> get_pins U1/*
{U1/A1 U1/A2 U1/A3 U1/A4}
```

The **create_port** command created ports in the *MID* design. To confirm that the ports are created, use the **get_pins** command on the pins of the instance of the *MID* design.

SEE ALSO

- `all_connected(2)`
- `connect_net(2)`
- `create_bus(2)`
- `current_design(2)`
- `disconnect_net(2)`
- `remove_bus(2)`
- `remove_port(2)`

create_power_domain

Creates a power domain, which provides a power supply distribution network.

SYNTAX

```
string create_power_domain
    domain_name
    [-elements cells]
    [-include_scope]
    [-scope instance_name]
    [-supply {supply_set_handle_name [supply_set_name]}]*
    [-available_supplies {supply_set_name_list}]
    [-update]
```

Data Types

```
domain_name      string
cells            collection
instance_name    string
supply_set_handle_name string
supply_set_name  string
supply_set_name_list collection
```

ARGUMENTS

domain_name

Specifies the name of the power domain to be created. It must be a simple (nonhierarchical) name.

The power domain cannot be created if there is an existing power domain, hierarchical cell instance, leaf cell instance, or port with the same name in the specified scope.

-elements cells

Specifies a collection of cells (hierarchical, black box, macro, buffer/inverter, or I/O pad cells) that are added as an extent of the power domain. The specified cells cannot be added to other power domains of the same scope.

If you do not use either the **-elements** or the **-include_scope** option, the power domain consists of the current scope and any of its children not specified as elements in another **create_power_domain** command.

-include_scope

Causes the scope of the power domain (the entire hierarchical level of the domain) to be included in the extent of the domain (the set of logic elements that belong to the power domain). This option is deprecated in IEEE Std. 1801-2013. Define the extent of the domain to include the current scope and, by default, all of its descendant scopes.

-scope *instance_name*

Specifies the scope (level of logic hierarchy) in which the power domain is created. The instance name is the name of a hierarchical cell. By default, the power domain is created in the current scope. This option is deprecated in IEEE Std. 1801-2013. Create the power domain within the current scope.

-supply {*supply_set_handle_name* *supply_set_name*}

Creates a user-specified supply set handle or restricts the supply sets available for use in the power domain. You can use this option multiple times in one command.

A supply set handle is an abstract supply set created for a power domain. By default, a power domain has supply set handles for the domain's primary supply set, a default isolation supply set, and a default retention supply set. These supply set handles let you synthesize a design even before you create any supply sets, supply nets, and supply ports for the power domain. Before such a design can be physically implemented, its supply set handles must be associated with actual supply sets. Those supply sets must be associated with actual supply nets.

If you specify a supply set handle name alone, without a supply set, the command creates a supply set handle having the specified name for the power domain. Supply nets named *domain_name.handle_name.power* and *domain_name.handle_name.ground* can then be used as supplies.

If you specify a supply set handle name with a supply set name, the command associates that supply set with the supply set handle, so that the specified supply set becomes equivalent to the supply set handle.

The following supply set handle names have special meaning in the command syntax:

```
primary
default_isolation
default_retention
extra_supplies_# (where # is a unique numeral)
extra_supplies
```

This is how the special-purpose supply set handle names are used:

- **primary**: The primary supply set of this power domain.
- **default_isolation**: The default isolation supply set for any isolation strategy applied to this power domain, if the isolation power or ground is not otherwise specified in this strategy.
- **default_retention**: The default retention supply set for any retention strategy applied to this power domain, if the retention power or ground is not otherwise specified in this strategy.
- **extra_supplies_#**: Extra supply sets that are available for the power domain to use in level shifter insertion and always-on synthesis.
- **extra_supplies**: When this supply set handle is used with an empty string for the *supply_set_name* parameter, **-supply {extra_supplies ""}**, it prevents any domain-independent extra supplies from being used in the power domain. In this case, only domain-dependent supply nets can be used, which are supply nets created with the **-domain** option of the **create_supply_net** command.

-available_supplies {*supply_set_name_list*}

A list of additional supply sets that are available for use by implementation tools to power cells inserted in this domain.

- The **-available_supplies** option specifies any additional supplies which are available for use.
- If **-available_supplies** is not specified, all supply sets and supply set handles defined in or above the scope of the power domain are available for use by tools to power cells inserted into the power domain.
- If **-available_supplies** is specified with an empty string argument, only the locally available supplies are available for use by tools to power cells inserted into the power domain.

- If **-available_supplies** is specified with a non-empty string, use a list of names of additional supply sets or supply set handles defined at or above the scope of the power domain in addition to the locally available supplies.

NOTE: The option **-available_supplies** cannot be used together with the keywords **extra_supplies** or **extra_supplies_#** of the **-supply** option.

-update

Updates the specification of an existing power domain. This option can be used only to update the supply set association or elements of a power domain, using the following other options of the command:

- **-supply**: Updates the supply set association of an existing power domain.
- **-available_supplies**: Updates the supply set association of an existing power domain.
- **-elements**: Updates the set of root cells that comprise the power domain extent.

The **-update** option is used along with the *domain_name* and **-supply** or **-elements** options. It is an error to use the **-update** option with any other options.

The **-supply** option allows you to

- Add a new supply set handle name without explicitly defining a supply set association
- Add a supply set to be associated with a previously declared supply set handle
- Add a new supply set handle and a supply set association in a single command

NOTE: You cannot update a supply set handle that you have already associated with a supply set.

The **-elements** option allows you to

- Add new root cells to an existing power domain extent

NOTE: You cannot add a root cell which is already a root cell of an existing power domain.

DESCRIPTION

This command creates a power domain in the specified scope. A power domain is a collection of design elements that share a primary power net and a ground power net. The logic hierarchy level where a power domain is created is called the *scope* of the power domain. The set of design elements that belong to a power domain is called the *extent* of that power domain. A hierarchical cell is an example of an element. Although a design element can be in the scope of several power domains, it can be in the extent of only one power domain.

A power domain can have several supply nets, which are connected to a power domain via a supply port. A power switch of a power domain can be used to turn on and off the power supply of part or all of the power domain. You can create a supply net, supply port, or power switch by using the **create_supply_net**, **create_supply_port**, or **create_power_switch** command, respectively.

When this command succeeds, it returns the full name of the power domain (from the current scope). When it fails, it returns a null string.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates two power domains in the scope named INST1.

```
prompt> create_power_domain PD1 -elements INST1/SUB_INST -scope INST1
INST1/PD1
prompt> create_power_domain PD2 -elements INST1/SUB_INST -scope INST1
Error: Could not add cell 'INST1/SUB_INST' to power domain 'PD2'.
(MWUI-402)
prompt> create_power_domain PD2 -scope INST1 -include_scope
INST1/PD2
```

The following example creates a power domain with the **-supply** option with the **extra_supplies_#** keyword.

```
prompt> create_power_domain PD_MID -scope mid1 \
-supply {extra_supplies_1 supply_set1} \
-supply {extra_supplies_2 supply_set1}
mid1/PD_MID
```

The following example creates a power domain with the **-supply** option with the **extra_supplies** keyword.

```
prompt> create_power_domain PD_MID -scope mid1 \
-supply {extra_supplies ""}
mid1/PD_MID
```

The following example updates an existing power domain with the **-supply** option.

```
prompt> create_power_domain PD_MID -scope mid1 -supply {abc}
mid1/PD_MID
prompt> create_power_domain PD_MID -scope mid1 -update \
-supply {primary SS1} -supply {abc SS1}
mid1/PD_MID
```

SEE ALSO

create_supply_set(2)
remove_power_domain(2)
report_power_domain(2)

create_power_state_group

Defines a group name to be used in the **add_power_state** command.

SYNTAX

```
status create_power_state_group  
      group_name
```

Data Types

group_name string

ARGUMENTS

group_name

Specifies the name of the group to create. It must use a simple name.

DESCRIPTION

The **create_power_state_group** command defines a group in the active scope that can be used with the **-group** option of the **add_power_state** command.

Use a group to collect related power states defined by the **add_power_state** command. The legal power states of a group define the legal combinations of power states of other objects in this scope or the descendant subtree, that is, those combinations of states of those objects that can exist at the same time during operation of the design.

It is an error if the specified group name already exists in the active scope.

The group name should be in the same namespace as the other UPF objects (power domains/supply sets/supply nets/supply ports/PSTs).

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to define a power state group.

```
prompt> create_power_state_group GROUP
```

SEE ALSO

`add_power_state(2)`

create_power_switch

Creates a power switch in the specified power domain.

SYNTAX

```
string create_power_switch
  switch_name
  -domain domain_name
  -output_supply_port {port_name supply_net_name}
  -input_supply_port {port_name supply_net_name}
  -control_port {port_name net_name}
  [-ack_port {port_name net_name [{boolean_function}]}]
  [-ack_delay {port_name delay}]
  -on_state {state_name input_supply_port {boolean_function}}
  [-off_state {state_name {boolean_function}}]
  [-on_partial_state {state_name {boolean_function}}]
  [-error_state {state_name {boolean_function}}]
  [-supply_set supply_set_reference]
```

Data Types

<i>switch_name</i>	string
<i>domain_name</i>	string
<i>port_name</i>	string
<i>supply_net_name</i>	string
<i>net_name</i>	string
<i>boolean_function</i>	list
<i>delay</i>	string
<i>state_name</i>	string
<i>input_supply_port</i>	string

ARGUMENTS

switch_name

Specifies the name of the power switch to be created. The name should be a simple (nonhierarchical) name. If a power switch with the same name already exists in the specified power domain, the power switch cannot be created.

This option is required.

-domain *domain_name*

Specifies the power domain that contains the power switch. If the power domain with the specified name does not exist in the current scope, the command fails.

This option is required.

-output_supply_port {*port_name* *supply_net_name*}

Specifies the name of the output port of the power switch and the supply net to which the port connects. On the power switch, if a port with the same name exists, the command fails. In the current scope, if there is no supply net with the same name, the command fails.

This option is required.

-input_supply_port {*port_name* *supply_net_name*}

Specifies the name of the input port of the power switch and the supply net to which the port connects. On the power switch, if a port exists with the same name, the command fails. In the current scope, if there is no supply net with the same name, the command fails.

This option is required and can be specified more than once. One power switch can have multiple input supply ports.

-control_port {*port_name* *net_name*}

Specifies the name of the control port of the power switch and the logical net to which this port connects. If a port with the specified name already exists on the power switch, the command fails. If a net with the specified name does not exist, the command fails.

This option is required and can be specified more than once. One power switch can have multiple control ports.

-ack_port {*port_name* *net_name* {*boolean_function*}}

Specifies the name of the acknowledge port of the power switch and the logical net to which this port connects. If a port with the specified name already exists on the power switch, the command fails. If a net with the specified name does not exist, the command fails.

If this option is not specified, the power switch does not have an acknowledge port. Optionally, a Boolean function can also be specified. The *boolean_function* should be defined in terms of the specified control ports, and nothing else.

This option can be specified multiple times.

-ack_delay {*port_name* *delay*}

Specifies the acknowledge port on the switch and the corresponding acknowledge delay.

This option can be specified multiple times.

-on_state {*state_name* *input_supply_port* {*boolean_function*}}

Specifies a named on-state, the relevant input supply port, and its Boolean function.

The *boolean_function* should be defined in terms of the specified control ports, and nothing else.

This option can be specified multiple times.

-off_state {*state_name* {*boolean_function*}}

Specifies a named off state and its relevant Boolean function.

The *boolean_function* should be defined in terms of the specified control ports, and nothing else.

This option can be specified multiple times.

-on_partial_state {*state_name* {*boolean_function*}}

Specifies a named partial_on state and its relevant Boolean function.

The *boolean_function* should be defined in terms of the specified control ports, and nothing else. The **-on_partial_state** option does not affect implementation and is intended to be used by simulation, so the tool reads and writes the information specified with this option transparently.

This option can be specified multiple times.

-error_state {state_name {boolean_function}}

Specifies a named error state and its relevant Boolean function.

The *boolean_function* should be defined in terms of the specified control ports, and nothing else. The **-error_state** option does not affect implementation and is intended to be used by simulation, so the tool reads and writes the information specified with this option transparently.

This option can be specified multiple times.

DESCRIPTION

The *create_power_switch* command enables you to create a power switch at the specified power domain. The switch is created within the scope of the power domain. Each power switch must be connected with an input supply net and an output supply net. The power switch can be connected with an acknowledge (logical) net and several control (logical) nets. Each net is connected with a power switch via a switch port. The switch ports are automatically created if the power switch is created successfully.

Upon success, this command returns the full name of the power switch (from the current scope). The command returns a null string upon failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates power switch *SW1* within the power domain *PD1*:

```
prompt> create_power_switch SW1 -domain PD1 \
  -output_supply_port {vout VNO2} \
  -input_supply_port {vin1 VNI1} \
  -input_supply_port {vin2 VNI2} \
  -control_port {ctrl_small ON1} \
  -control_port {ctrl_large ON2} \
  -ack_port {ack_p ACKN {ctrl_small & !ctrl_large}} \
  -ack_delay {ack_p 1} \
  -on_state {full_s vin1 {ctr_small}} \
  -off_state {off_s {!ctr_small}}
```

SEE ALSO

report_power_switch(2)

create_pst

Creates a power state table (PST), using a specific order of supply nets.

SYNTAX

```
string create_pst  
    table_name  
    -supplies list
```

Data Types

```
table_name  string  
list        list
```

ARGUMENTS

table_name

Specifies the name of the power state table.

This argument is required.

-supplies *list*

Specifies a list of supply nets or ports to include in each power state table in the design.

This option is required.

DESCRIPTION

The **create_pst** command creates a power state table using a specific order of supply nets. A power state table is used for implementation specifically for synthesis, analysis, and optimization. The power state table defines the legal combinations of states, which are those combinations of states that can exist at the same time during the operation of the design.

The power state table has no simulation semantics. It is tool-dependent as to whether the simulation tools report an error if an illegal (unspecified) combination of states occurs.

An error occurs if a specified supply net has not been created before the **create_pst** command is run.

This command returns the name of the power state table if it is created, or the null string if the power state table is not created.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of the **create_pst** command:

```
prompt> create_pst MyPowerStateTable -supplies {PN1 PN2 SOC/OTC/PN3}
```

SEE ALSO

add_port_state(2)
add_pst_state(2)
report_pst(2)

create_qor_snapshot

Generates a quality-of-results report for active scenarios of the current design and stores the report into a set of files.

SYNTAX

```
status create_qor_snapshot
  -name name
  [-power]
  [-clock_tree]
  [-show_all]
  [-route]
  [-save_mw]
  [-significant_digits digits]
  [-zero_wire_load]
  [-max_paths number]
  [-nworst number]
  [-scenarios list_of_scenarios]
  [-infeasible_paths]
  [-nosplit]
```

Data Types

<i>name</i>	string
<i>digits</i>	integer
<i>number</i>	integer
<i>list_of_scenarios</i>	string

ARGUMENTS

-name *name*

Specifies the name of the QoR snapshot, which can be used to identify a particular snapshot. The name should start with a letter.

-power

Records dynamic and static power.

-clock_tree

Records clock tree information. This option is supported only in `icc_shell`, not `dc_shell` or `de_shell`.

-show_all

Shows all clock groups, including those without violations. By default, the command only shows clock groups that have violations.

-route

Shows route data of the design. This option is supported only in `icc_shell`, not `dc_shell` or `de_shell`.

-save_mw

Saves the Milkyway design using the snapshot name appended with "_ss". This option is supported only in `icc_shell`, not `dc_shell` or `de_shell`.

-significant_digits *digits*

Specifies the number of significant digits for the values generated in the reports. This setting can affect the numbers of violating paths in the histogram of minimum and maximum violations might, as demonstrated in the EXAMPLES section.

-zero_wire_load

Uses zero wire load models for timing reporting.

-max_paths *number*

Specifies the maximum number of paths for the **report_timing** command to report per path group.

-nworst *number*

Specifies the maximum number of worst paths for the **report_timing** command to report per timing endpoint.

-scenarios *list_of_scenarios*

Specifies the list of active scenarios to get a QoR snapshot. The default behavior is to report all active scenarios.

-infeasible_paths

Stores the data of paths which have been marked as infeasible during compile exploration. To access the stored data, specify the **-infeasible_paths** option with the **query_qor_snapshot** command.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **create_qor_snapshot** command generates a quality-of-results report for the active scenarios of the current design and stores the report into a set of files. You can extract information from snapshot reports by using the **query_qor_snapshot** command. A QoR snapshot contains information about quality metrics such as timing, area, DRC, clocks, power, and routing.

When you use the **-zero_wire_load** option, the timing report is generated using zero interconnect delay mode. The timing report is also affected by the **set_zero_interconnect_delay_mode** command.

The command supports both multicorner-multimode designs as well as designs that are not multicorner-multimode.

Multicorner-Multimode Support

By default, this command uses information from all active scenarios. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following command creates a QoR snapshot named "preroute" from the current design and saves the snapshot files under the "snapshot" directory under in the current working directory.

```
prompt> create_qor_snapshot -name preroute
```

The following example shows how the number of significant digits can affect the output of the minimum and maximum violation histogram.

```
prompt> create_qor_snapshot -name atest -significant_digits 4
```

```
...
-----
Histogram:      s1
-----
Max violations:  0
above ~ -0.7 --- 0
-0.6 ~ -0.7 --- 0
-0.5 ~ -0.6 --- 0
-0.4 ~ -0.5 --- 0
-0.3 ~ -0.4 --- 0
-0.2 ~ -0.3 --- 0
-0.1 ~ -0.2 --- 0
0 ~ -0.1 --- 0
-----
Min violations:  473
-0.06 ~ above --- 0
-0.05 ~ -0.06 --- 0
-0.04 ~ -0.05 --- 0
-0.03 ~ -0.04 --- 0
-0.02 ~ -0.03 --- 2
-0.01 ~ -0.02 --- 34
0 ~ -0.01 --- 437
-----
...
```

```
prompt> create_qor_snapshot -name atest -significant_digits 3
```

```
...
-----
Histogram:      s1
-----
Max violations:  0
above ~ -0.7 --- 0
-0.6 ~ -0.7 --- 0
-0.5 ~ -0.6 --- 0
-0.4 ~ -0.5 --- 0
-0.3 ~ -0.4 --- 0
-0.2 ~ -0.3 --- 0
-0.1 ~ -0.2 --- 0
0 ~ -0.1 --- 0
-----
Min violations:  473
-0.06 ~ above --- 0
-0.05 ~ -0.06 --- 0
-0.04 ~ -0.05 --- 0
-0.03 ~ -0.04 --- 0
```

```
-0.02 ~ -0.03 --- 2  
-0.01 ~ -0.02 --- 42  
0 ~ -0.01 --- 429  
-----
```

SEE ALSO

query_qor_snapshot(2)
set_zero_interconnect_delay_mode(2)

create_qtm_clock

Creates a quick timing model (QTM) clock port.

SYNTAX

```
status create_qtm_clock  
-type clock | generated_clock  
clock_list
```

Data Types

clock_list list

ARGUMENTS

-type clock | generated_clock

Specifies the type of clock. The clock type helps in defining an existing port or creating an internal pin as a clock port.

clock_list

Specifies the list of QTM clock ports that you want to create.

DESCRIPTION

This command creates a QTM clock port. You can create a single clock port or a list of clock ports with this command. If there is an existing QTM port with the same port name, the clock port has the same direction as the previously created port. If there no existing QTM port with the same port name, a new port definition is created with the direction as internal.

Note that this command does not create a clock definition. It just creates an internal clock pin or designates an existing QTM port as a clock port.

To show the information about the current QTM model, use the **report_qtm_model** command.

For a basic description about QTMs, see the **create_qtm_model** man page. For a more detailed description about QTMs, see the *IC Compiler Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example creates a QTM clock port named CLK.

```
prompt> create_qtm_clock CLK -type clock
```

The following example creates QTM output clock port named C.

```
prompt> create_qtm_port {C} -type output  
prompt> create_qtm_clock {C} -type clock
```

The following example creates QTM internal generated clock pin named C_int.

```
prompt> create_qtm_clock C_int -type generated_clock
```

SEE ALSO

- `create_qtm_model(2)`
- `report_qtm_model(2)`
- `save_qtm_model(2)`
- `set_qtm_port_drive(2)`
- `set_qtm_port_load(2)`

create_qtm_constraint_arc

Creates a constraint arc for a quick timing model (QTM).

SYNTAX

```
status create_qtm_constraint_arc
  [-name arc_name]
  -setup | -hold
  -from port_name
  -to port_spec
  -edge rise | fall
  [-signal_edge rise | fall]
  [-input_transition_rise rise_time]
  [-input_transition_fall fall_time]
  -path_type name [-path_factor factor]
  | -value constraint_value
```

Data Types

<i>arc_name</i>	string
<i>port_name</i>	string
<i>port_spec</i>	list
<i>rise_time</i>	float
<i>fall_time</i>	float
<i>name</i>	string
<i>factor</i>	float
<i>constraint_value</i>	float

ARGUMENTS

-name *arc_name*

Specifies the name of the constraint arc.

If you do not specify the **-name** option, the command constructs a default name based on the port names and edge names.

-setup

Creates a setup arc.

This option and the **-hold** option are mutually exclusive; you must specify one.

-hold

Creates a hold arc.

This option and the **-setup** option are mutually exclusive; you must specify one.

-from *port_name*

Specifies the constraining port name. Define this port as a clock port.

This is a required option.

-to *port_spec*

Specifies the constrained port. Define this port as an input, inout, or internal port.

This is a required option.

-edge *rise* | *fall*

Specifies the triggering edge of the clock specified by the **-from** option.

This is a required option.

-signal_edge *rise* | *fall*

Specifies the direction of the signal at the constrained port specified by the **-to** option.

If you do not specify this option, the tool creates constraints for both signal rise and signal fall.

-input_transition_rise *rise_time*

Specifies the rise transition time used for delay calculation at the input pin specified by the **-to** option.

This option works together with the **setup** or **hold** global QTM parameter specified by the **set_qtm_global_parameter** command, depending on whether you are specifying a setup or hold constraint. This delay and the delay specified by the **-value** option are applied to the arc. The **-input_transition_rise** option is ignored if there is no global setup or hold parameter specified by the **set_qtm_global_parameter** command.

The default is 0.0.

-input_transition_fall *fall_time*

Specifies the fall transition time used for delay calculation at the input pin specified by the **-to** option.

This option works together with the **setup** or **hold** global QTM parameter specified by the **set_qtm_global_parameter** command, depending on whether you are specifying a setup or hold constraint. This delay and the delay specified by the **-value** option are applied to the arc. The **-input_transition_fall** option is ignored if there is no global setup or hold parameter specified by the **set_qtm_global_parameter** command.

The default is 0.0.

-path_type *name*

Specifies the path type to use to set the delay. The specified type must be defined by a previous call to the **create_qtm_path_type** command.

This option and the **-value** option are mutually exclusive; you must specify one.

-path_factor *factor*

Specifies the multiplication factor for the path type.

This option is valid only with the **-path_type** option.

-value *constraint_value*

Specifies the delay value in the time units used in the design.

This option and the **-path_type** option are mutually exclusive; you must specify one.

DESCRIPTION

This command creates a constraint arc in a quick timing model. The **-from** option specifies the constraining port and the **-to** option specifies the constrained port. The **-from** port must be defined as a clock port and the **-to** port must be an input, inout, or internal port.

If the constraint arc is a setup arc, the tool adds the QTM global setup time to the value specified by the **-setup** option. If the constraint arc is a hold arc, the tool adds the QTM global hold time to the value specified by the **-hold** option. You define the global setup time and global hold time by using the **set_qtm_global_parameter** command.

To display information about the current QTM model, use the **report_qtm_model** command.

For a basic description of quick timing models, see the **create_qtm_model** man page. For a more detailed description, see the *IC Compiler Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command creates a setup arc from the positive edge of the constraining pin CLK to the constrained pin IN1. The delay is 2 times the delay for path1.

```
prompt> create_qtm_constraint_arc -setup -edge rise \  
-from CLK -to IN1 -path_type path1 -path_factor 2
```

The following command creates a hold arc from the positive edge of the constraining pin CLK to the constrained pin IN1. The delay is 2 times the delay for path1.

```
prompt> create_qtm_constraint_arc -hold -edge rise \  
-from CLK -to IN1 -path_type path1 -path_factor 2
```

The following command creates a hold arc from the positive edge of the constraining pin CLK to the constrained pin IN1. The delay is 3 time units.

```
prompt> create_qtm_constraint_arc -hold -edge rise \  
-from CLK -to IN1 -value 3.0
```

SEE ALSO

create_qtm_delay_arc(2)
create_qtm_model(2)
create_qtm_path_type(2)

```
report_qtm_model(2)  
save_qtm_model(2)  
set_qtm_global_parameter(2)
```

create_qtm_delay_arc

Creates a delay arc for a quick timing model (QTM).

SYNTAX

```
status create_qtm_delay_arc
  [-name arc_name]
  -from ports
  -to ports
  [-from_edge rise | fall]
  [-to_edge rise | fall]
  [-input_transition_rise rise_time]
  [-input_transition_fall fall_time]
  -path_type name [-path_factor factor]
  | -value delay_value
  | -total_value total_value
  [-max_insertion_delay]
  [-min_insertion_delay]
```

Data Types

```
arc_name    string
ports       collection
rise_time   float
fall_time   float
name        string
factor       float
delay_value float
total_value float
```

ARGUMENTS

-name *arc_name*

Specifies the name of the delay arc.

If you do not specify the **-name** option, the command constructs a default name based on the port names and edge names.

-from *ports*

Specifies the QTM ports that are the startpoints of the delay arc. The port type must be clock, input, inout, or internal.

-to *ports*

Specifies the QTM ports that are the endpoints of the delay arc. The port type must be output, inout, or internal.

-from_edge *rise | fall*

Specifies the triggering edge (rise or fall) for clock startpoints or the signal direction for data startpoints. If you do not use this option, the command creates delay arcs from both rise and fall edges at the startpoints.

-to_edge *rise | fall*

Specifies the signal direction at the data endpoints. If you do not use this option, the command creates delay arcs to both rise and fall edges at the endpoints.

If you omit both the **-from_edge** and **-to_edge** options, the command creates rise-to-rise and fall-to-fall delay arcs.

-input_transition_rise *rise_time*

Specifies the rise transition time used for delay calculation at the input pin specified by the **-to** option. The default is 0.0.

-input_transition_fall *fall_time*

Specifies the fall transition time used for delay calculation at the input pin specified by the **-to** option. The default is 0.0.

When you specify a delay arc from a clock port, the **-input_transition_rise** or **-input_transition_fall** option works together with the **clk_to_output** global QTM parameter specified by the **set_qtm_global_parameter** command. This delay and the delay specified by the **-value** or **-total_value** option are applied to the arc. If there is no global **clk_to_output** parameter specified by the **set_qtm_global_parameter** command, the **-input_transition_rise** option or **-input_transition_fall** option setting is ignored.

-path_type *name*

Specifies the name of the path type to use to set the delay. The specified type must be defined by previous usage of the **create_qtm_path_type** command.

The **-path_type**, **-value**, and **-total_value** options are mutually exclusive; you must use one of them.

-path_factor *factor*

Specifies the multiplication factor for the path type. The delay value for this path is the delay of the specified path type multiplied by this factor. This option is used only with the **-path_type** option.

-value *delay_value*

Specifies the delay value in terms of the time units used for the design. The total delay value for the arc is computed by adding the specified value to the delay computed for the drive arc at the output port. The drive arc delay is defined by the **create_qtm_drive_type** and **set_qtm_port_drive** commands.

-total_value *total_value*

Specifies the total delay value in terms of the time units used for the design. The drive arc delay and load are not considered when determining the total delay value for the arc.

-max_insertion_delay

Specifies the insertion delay value for the maximum operating condition.

-min_insertion_delay

Specifies the insertion delay value for the minimum operating condition.

When you specify the **-max_insertion_delay** or **-min_insertion_delay** option, the **-from** and **-to** option must be the same port and must be a clock port. When you use these options, the command creates the insertion delay on that clock port of QTM model.

The **-max_insertion_delay** and **-min_insertion_delay** options must be specified with the **-value** option. In addition, these options are mutually exclusive with the **-path_type**, **-path_factor**, **-total_value**, **-from_edge**, **-to_edge**, **-name**, **-input_transition_rise**, and **-input_transition_fall** options.

DESCRIPTION

This command creates delay arcs in a quick timing model. If you specify a list of ports for the **-from** and **-to** options, the tool creates delay arcs from each startpoint to each endpoint.

For an edge-triggering arc from a clock port to a data port, the value of the **clk_to_output** global QTM parameter is added to the specified delay value.

You can specify the delay value of the arc by specifying a path type, by specifying a value that is added to the computed delay for the drive arc, or by specifying a total delay value explicitly. You must use exactly one of the options **-path_type**, **-value**, or **-total_value** to specify the delay value.

If the delay defined by **create_qtm_delay_arc** for the port is the maximum delay budgeted for the output, use the **-total_value** option to prevent the command from adding any more delay. If you use the **-value** option, the drive arc delay might be added to the budget. Note that with a smaller load at the output port, the delay might be smaller than the budgeted maximum delay.

To show the information for the current quick timing model, use the **report_qtm_model** command.

For a basic description of quick timing models, see the **create_qtm_model** man page. For a more detailed description, see the *IC Compiler Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command creates rise-to-fall delay arcs from the A, B, and C input ports to output port D with a delay value of 2.0 time units. The total delay for each arc is computed by adding 2.0 to the drive arc delay.

```
prompt> create_qtm_delay_arc -from_edge rise -to_edge fall \
    -from {A B C} -to D -value 2.0
```

The following command creates rise-to-rise and fall-to-fall delay arcs from the clock port named CLK to the OUT output port with a delay equivalent to three times that of path type "path1", which was defined earlier by the **create_qtm_path_type** command.

```
prompt> create_qtm_delay_arc -from CLK -to OUT \
    -path_type path1 -path_factor 3
```

The following command uses a wildcard to create rise-to-rise and fall-to-fall delay arcs from all input ports whose names start with CL to the OUT output port, with a delay equivalent to three times that of path type "path1".

```
prompt> create_qtm_delay_arc -from CL* -to OUT \
    -path_type path1 -path_factor 3
```

The following command sets the insertion delay on the clock port named CLK to be 4.5 for both the maximum and minimum operating conditions.

```
prompt> create_qtm_delay_arc -from CLK -to CLK -value 4.5 \
    -max_insertion_delay -min_insertion_delay
```

SEE ALSO

`create_qtm_constraint_arc(2)`
`create_qtm_model(2)`
`create_qtm_path_type(2)`
`report_qtm_model(2)`
`save_qtm_model(2)`

create_qtm_drive_type

Creates a drive type in a quick timing model (QTM).

SYNTAX

```
status create_qtm_drive_type
-lib_cell lib_cell_name
[-input_pin pin_name]
[-output_pin pin_name]
[-input_transition_rise rtrans]
[-input_transition_fall ftrans]
drive_type_name
```

Data Types

<i>lib_cell_name</i>	string
<i>rtrans</i>	float
<i>ftrans</i>	float
<i>drive_type_name</i>	string

ARGUMENTS

-lib_cell *lib_cell_name*

Specifies the library cell name in the technology library.

This is a required option.

-input_pin *pin_name*

Specifies the input pin in the lib_cell that is the startpoint of the arc that ends in the output pin.

-output_pin *pin_name*

Specifies the output pin in the lib_cell that specifies the drive.

-input_transition_rise *rtrans*

Specifies the input rising transition time associated with the input pin to compute the delay for the drive arc.

If you do not use this option, the delay table for the rising input of the drive arc is loaded from the library as-is.

Use the **-input_transition_rise** and **-input_transition_fall** options to capture the transition time associated with the input pin. This can provide more accurate information on the transition time and delay time for the defined drive arc.

-input_transition_fall *ftrans*

Specifies the input falling transition time associated with the input pin to compute the delay for the drive arc.

If you do not use this option, the delay table for the falling input of the drive arc will be loaded from library as-is.

drive_type_name

Specifies the name given to the defined drive type.

DESCRIPTION

This command can be used to create a drive type. The defined drive type can be used to set the drives on output ports. The drive type is specified by referring to a library cell in the technology library. The library cell is specified by using the **-lib_cell** option. This library cell must be present in the technology library.

The output pin that drives the port is specified by using the **-output_pin** option. In addition, you can specify the input pin by using the **-input_pin** option. This option selects an arc that drives the output pin of the library cell.

If you do not specify the input pin or the output pin, the tool chooses an arbitrary arc to define the drive type. If you specify the input pin but not the output pin, the tool chooses an arbitrary arc starting from the input pin to define the drive type. If you specify the output pin but not the input pin, the tool chooses an arbitrary delay arc coming into the output pin to define the drive type. If you specify both the input and the output pin, the library cell must have a combinational arc from the input pin to the output pin.

To use this command, you must read in the technology library and then specify it by using the **set_qtm_technology** command.

To show the information about the current QTM model, use the **report_qtm_model** command.

For a basic description about QTMs, see the **create_qtm_model** man page. For a more detailed description, see the *IC Compiler Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

In the following examples it is assumed that you have loaded the technology library and the QTM technology is set. Do this by using the following the sequence of commands as a guide, substituting the name of your technology library.

```
prompt>read_lib my_technology_library.db
```

```
prompt>set_qtm_technology -library my_technology_library
```

The following command creates a drive type equivalent to the BUF1 library cell.

```
prompt>create_qtm_drive_type -lib_cell BUF1 drive1
```

The following command creates a drive type named drive2 that is equivalent to the BUF2 library cell and specifies the output pin to use.

```
prompt>create_qtm_drive_type -lib_cell BUF2 -output_pin Y drive2
```

SEE ALSO

create_qtm_load_type(2)
create_qtm_model(2)
create_qtm_path_type(2)
report_qtm_model(2)
save_qtm_model(2)
set_qtm_port_drive(2)
set_qtm_technology(2)

create_qtm_generated_clock

Creates a generated clock for a quick timing model (QTM).

SYNTAX

```
status create_qtm_generated_clock  
-source master_clock_name  
[-divide_by divide_factor  
 | -multiply_by multiply_factor]  
[-invert]  
generated_clock_name
```

Data Types

```
master_clock_name    string  
divide_factor        int  
multiply_factor      int  
generated_clock_name string
```

ARGUMENTS

-source *master_clock_name*

Specifies the name of the clock defined as the master source of the generated clock. The master source must be defined as a clock or a generated clock prior to the generated clock definition.

-divide_by *divide_factor*

Specifies the frequency division factor. If the *divide_factor* value is 2, the generated clock period is twice as long as the master clock period.

-multiply_by *multiply_factor*

Specifies the frequency multiplication factor. If the *multiply_factor* value is 3, the generated clock period is one-third as long as the master clock period.

-invert

Inverts the generated clock signal (in the case of frequency multiplication and division).

generated_clock_name

Specifies the name of the generated clock.

If a QTM port or internal pin with the specified name already exists, the generated clock is defined on the existing port or internal pin. Otherwise, a new internal pin is created to be the source of the generated clock.

DESCRIPTION

This command creates a QTM generated clock. A generated clock can be defined on an external port of the QTM, as well as on an internal pin in the QTM. After you define a generated clock, the source port or pin can be used in other QTM commands the same way as those ports and pins defined by the **create_qtm_port** command. For example, the delay arcs from or to the generated clock source pin can be defined with the **create_qtm_delay_arc** command and constraint arcs related to the generated clock can be defined with the **create_qtm_constraint_arc** command.

For a basic description about QTMs, see the **create_qtm_model** man page. For a more detailed description about QTMs, see the *IC Compiler Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example creates a divide-by 2 generated clock using the master clock CLK on an internal pin named GCLK_int (assuming that there is no QTM port named GCLK_int).

```
prompt> create_qtm_generated_clock GCLK_int -source CLK -multiply_by 2
```

The following example creates a generated clock on port GCLK (assuming that GCLK has already been defined as a QTM port).

```
prompt> create_qtm_generated_clock GCLK -source CLK -divide_by 2 -invert
```

SEE ALSO

- create_qtm_model(2)
- report_qtm_model(2)
- save_qtm_model(2)
- create_qtm_delay_arc(2)
- create_qtm_constraint_arc(2)

create_qtm_insertion_delay

Specifies the insertion delay on the clock port of a quick timing model (QTM).

SYNTAX

```
string create_qtm_insertion_delay  
  [-max]  
  [-min]  
  [-value insertion_delay]  
  port_list
```

Data Types

```
insertion_delay  float  
port_list        list
```

ARGUMENTS

-max

Specifies whether to generate insertion delay for Maximum operating condition.

-min

Specifies whether to generate insertion delay for Minimum operating condition.

-value *insertion_delay*

Specifies the insertion delay in terms of the time units you use for the model.

port_list

Specifies a list of clock ports on which to set internal clock latency. Each element in the list is either a collection of QTM ports or a pattern that matches QTM ports.

DESCRIPTION

This command creates the insertion delay for a clock port on the QTM model. The insertion delay is the network latency from the clock port to a (hypothetical) FlipFlop inside the QTM model.

If the CTS has been done on the design, then there would be network latency for all the clock pins of the FlipFlops. If some of the hierarchical blocks in the design are replaced by blackboxes, then the user can generate QTM models for the blackboxes to

represent their timing model.

If the user generates the QTM models for the blackboxes, then `create_qtm_insertion_delay` command can be called to set the network delay from clock port of the hierarchical block to FlipFlop inside the hierarchical block, as the insertion delay.

To show the information about the current QTM model, use the **report_qtm_model** command.

For a basic description about QTM, see the **create_qtm_model** man page. For a more detailed description about QTM, see the design planning chapter of the *IC Compiler User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command sets the insertion delay on CLK to be 2.0 for Max mode.

```
prompt> create_qtm_insertion_delay -max -value 2.0 CLK
```

The following command sets the insertion on CK to be 4.5 for both Max and Min modes.

```
prompt> create_qtm_insertion_delay -value 4.5 CK
```

SEE ALSO

- `create_qtm_model(2)`
- `create_qtm_port(2)`
- `report_qtm_model(2)`
- `save_qtm_model(2)`
- `create_qtm_clock(2)`

create_qtm_load_type

Creates a load type for a quick timing model (QTM) description.

SYNTAX

```
string create_qtm_load_type  
  -lib_cell lib_cell_name  
  [-input_pin pin_name]  
  load_type_name
```

Data Types

```
lib_cell_name  string  
pin_name      string  
load_type_name string
```

ARGUMENTS

-lib_cell *lib_cell_name*

Specifies the name of the library cell in the technology library.

-input_pin *pin_name*

Specifies the input pin in the lib_cell to specify capacitance.

load_type_name

Specifies the name given to the defined load type.

DESCRIPTION

This command creates a load type. A load type can be used to define the load on a QTM input port. The load type is specified by referring to a library cell in the technology library. The cell is specified using the **-lib_cell** option. The cell must be present in the specified technology library.

The input pin can be specified by using the **-input_pin** option. If the input pin is not specified, an arbitrary input pin is chosen to define the load type.

To use this command you must first read in the technology library, then specify the technology library by using the **set_qtm_technology** command.

To display information for the current QTM model, use the **report_qtm_model** command.

For a basic description about QTMs, see the **create_qtm_model** man page. For a more detailed description about QTM, see the *IC Compiler Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example loads the technology library and sets the QTM technology.

```
prompt>read_db my_technology_library.db
prompt>set_qtm_technology -library my_technology_library
```

The following command creates a load type 'load1' using cell OR1. Since the input pin name is not specified, QTM selects an arbitrary input pin.

```
prompt> create_qtm_load_type -lib_cell OR1 load1
```

The following command creates a load type 'load2' using cell OR1, input pin A.

```
prompt> create_qtm_load_type -lib_cell OR1 -input_pin A load2
```

SEE ALSO

```
create_qtm_drive_type(2)
create_qtm_model(2)
create_qtm_path_type(2)
report_qtm_model(2)
save_qtm_model(2)
set_qtm_port_load(2)
```

create_qtm_model

Begins the definition of a quick timing model (QTM) description.

SYNTAX

```
string create_qtm_model  
    model_name
```

Data Types

```
model_name  string
```

ARGUMENTS

model_name

Specifies the name of the generated model.

DESCRIPTION

Specifies the name of a new quick timing model (QTM) to be created by ICC Design Planning. QTMs are temporary timing models that can be created quickly for a block using ICC Design Planning commands. The purpose of these commands is to provide timing information without the necessity of writing a detailed timing model.

QTMs are intended to be used early in the design cycle to describe rough initial timing of a block. These models will eventually be replaced, either by some other detailed timing model or by the netlist of the block, to obtain more accurate timing. For a more detailed description of QTMs, see the *ICC Design Planning User Guide*.

A QTM model can be saved as a Synopsys DB file, which can be instantiated in a design in the same way library cells or ITS models are instantiated. Both Design Compiler and JupiterXT accept designs with instantiated QTMs. To save a QTM model, use the **save_qtm_model** command.

Other commands in the QTM command set must be placed between a **create_qtm_model** and **save_qtm_model** command. Many QTM commands exist for specifying, configuring and examining QTMs. The following is not an exhaustive list, but gives examples of tasks you can perform using QTM commands:

- To create a QTM port, use the **create_qtm_port** command.
- To create constraint arcs and delay arcs, use the **create_qtm_constraint_arc** and the **create_qtm_delay_arc** commands, respectively.
- To set the drive of a QTM output port, use the **set_qtm_port_drive** command.

- To set the load of a QTM input port, use **set_qtm_port_load** command.
- To show the information about the current QTM model, use the **report_qtm_model** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command creates a new QTM model with the name adder.

```
prompt> create_qtm_model adder
```

SEE ALSO

```
create_qtm_constraint_arc(2)
create_qtm_delay_arc(2)
create_qtm_drive_type(2)
create_qtm_load_type(2)
create_qtm_path_type(2)
create_qtm_port(2)
report_qtm_model(2)
save_qtm_model(2)
set_qtm_global_parameter(2)
set_qtm_port_drive(2)
set_qtm_port_load(2)
set_qtm_technology(2)
```

create_qtm_path_type

Creates a path type in a quick timing model (QTM) description.

SYNTAX

```
string create_qtm_path_type  
  -lib_cell lib_cell_name  
  [-input_pin input_pin_name]  
  [-output_pin output_pin_name]  
  [-fanout count]  
  path_type_name
```

Data Types

```
lib_cell_name    string  
input_pin_name  string  
output_pin_name string  
count           integer  
path_type_name string
```

ARGUMENTS

-lib_cell *lib_cell_name*

Specifies the name of the library cell in the technology library.

-input_pin *input_pin_name*

Specifies the input pin in the lib_cell. This is the startpoint of the arc for which to define the path type.

-output_pin *output_pin_name*

Specifies the output pin in the lib_cell. This is the endpoint of the arc for which to define the path type.

-fanout *count*

Specifies the average number of fanout pins to consider while computing the delay of the path type. The fanout count can range from 1 to 1000. By default, this is 1.

path_type_name

Specifies the name given to the defined path type.

DESCRIPTION

This command creates a path type. A path type mimics an arc in a library cell. An arc in the QTM model is defined in terms of the delays of the path type. The path type is specified by referring to a `lib_cell` in the technology library. The `lib_cell` is specified using the **-lib_cell** option. This `lib_cell` must be present in the technology library specified.

You can specify the average fanout count to which the arc fans out while defining the path type. It is assumed the arc fans out to the first input pin of the same library cell, while calculating the delays. The input pin is specified by using the **-input_pin** option. The output pin is specified by using the **-output_pin** option.

If neither input pin nor the output pin is specified, an arbitrary delay arc in the `lib_cell` is chosen to define the path type.

If the input pin is specified and output pin is not specified, an arbitrary delay arc starting from the input pin defines the path type.

If the output pin is specified and input pin is not specified, an arbitrary delay arc coming into the output pin defines the path type.

If both input and output pins are specified, the `lib_cell` must have an arc from the input pin to the output pin.

To use this command, you must read in the technology library. Next, specify the technology library by using the **set_qtm_technology** command.

To show information about the current QTM model, use the **report_qtm_model** command.

For a basic description about QTM, see the **create_qtm_model** man page. For a more detailed description about QTM, see the *IC Compiler Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following examples load the technology library and set the QTM technology.

```
prompt>read_lib my_technology_library.db
prompt>set_qtm_technology -library my_technology_library
```

The following example creates a path type `path1` by using `lib_cell AN2`, with an average fanout count of 2 (uses default input, output pins).

```
prompt> create_qtm_path_type -lib_cell AN2 -fanout 2 path1
```

The following example creates a path type `path2` by using cell `AN2`, with an average fanout count of 2, and using pin 'A' as the starting point for the path type.

```
prompt> create_qtm_path_type -lib_cell AN2 -input_pin A -fanout 2 path2
```

The following example creates a path type `path3` by using cell `AN2`, with an average fanout count of 2, using 'A' as the startpoint of the arc, and pin 'Z' as the endpoint of the arc.

```
prompt> create_qtm_path_type -lib_cell AN2 -input_pin A -output_pin Z -fanout 2 path3
```

SEE ALSO

`create_qtm_constraint_arc(2)`
`create_qtm_delay_arc(2)`
`create_qtm_drive_type(2)`
`create_qtm_load_type(2)`
`create_qtm_model(2)`
`report_qtm_model(2)`
`save_qtm_model(2)`

create_qtm_port

Creates a quick timing model (QTM) port.

SYNTAX

```
string create_qtm_port  
-type port_type  
port_list
```

Data Types

```
port_type  string  
port_list  list
```

ARGUMENTS

-type *port_type*

Specifies the type of port. The port can be one of the following types: input, output, inout, internal or clock. If you want a port to be a clock, define it as a clock port.

port_list

Specifies the list of QTM ports you want created.

DESCRIPTION

This command creates QTM port. You can create a single port or a list of ports with this command. The ports can be input, output, inout, internal or clock. Any port that constrains another port or has a edge triggered launch arc originating from it, has to be defined to be of the type 'clock'.

Bused ports are also defined by giving the start and end index (A[0:5]). Bused ports cannot be internal.

To show the information about the current QTM model, use **report_qtm_model** command.

For a basic description about QTM, please refer to **create_qtm_model** man page. For a more detailed description about QTM, please refer to the *ICC Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example creates an input bused QTM port A[0:5].

```
prompt> create_qtm_port A[0:5] -type input
```

The following example creates a clocked QTM port CLK.

```
prompt> create_qtm_port CLK -type clock
```

The following example creates QTM output ports A, B, C, and D.

```
prompt> create_qtm_port {A B C D} -type output
```

The following example creates QTM internal pin D_int.

```
prompt> create_qtm_port D_int -type internal
```

SEE ALSO

```
create_qtm_model(2)  
report_qtm_model(2)  
save_qtm_model(2)  
set_qtm_port_drive(2)  
set_qtm_port_load(2)
```

create_register_bank

Creates a multibit register bank from a list of single-bit registers.

SYNTAX

```
status create_register_bank  
  object_list  
  [-name bank_name]  
  -lib_cell library_name/lib_cell_name
```

Data Types

```
object_list  list  
bank_name    string
```

ARGUMENTS

object_list

Specifies a list of single-bit registers or latches in the current design from which a multibit register bank is created. The specified registers or latches are removed from the netlist and replaced by the multibit register bank. The order in which the registers are listed determines the order in which the nets are connected to the inserted register bank.

-name *bank_name*

Specifies the name of the new multibit register bank. If this option is omitted, the command derives a bank name by concatenating the names of *object_list*.

-lib_cell *library_name/lib_cell_name*

Specifies the name of the library reference cell used for the new register bank. The bit-width of the specified library cell must be at least as large as the total bit-width of the registers specified in the *object_list*. You can specify only one library cell name. Specifying the library name along with the cell name (for example, my_lib/my_cell) is mandatory in the Design Compiler tool and optional in the IC Compiler too.

DESCRIPTION

The **create_register_bank** command creates a new multibit bank from a list of registers or latches in the current design. All the single-bit registers or latches in *object_list* are replaced by one bank.

The command verifies that the registers in the *object_list* exist in the design and have valid locations, and do not have a "dont_touch" or "fixed" attribute.

The order of the specified registers determines the pin connection order of the new bank. For example, a net connected to the third specified register in the *object_list* is connected to the third bit of the multibit register inserted by the command.

In the Design Compiler tool, the bank bit-width has to be equal to the total bit-width of the specified registers.

To automatically generate a script containing **create_register_bank** bank commands based on specified banking criteria, use the **identify_register_banks** command in the Design Compiler tool or the **set_banking_guidance_strategy** command in the IC Compiler tool.

In the Design Compiler tool, you can use the **create_register_bank** and **split_register_bank** commands only before scan insertion. If you want to use these commands after scan insertion, use them later in the IC Compiler tool.

Multicorner-Multimode Support

In case of designs with multiple scenarios, to properly transfer scenario-specific information during banking operations, you need to activate all scenarios before you run the **create_register_bank** commands .

EXAMPLES

The following example creates a multibit register bank from 4 single-bit registers:

```
prompt> create_register_bank -name group0_0 \  
      {reg_1 reg_2 reg_3 reg_4} -lib_cell multibit_lib/REG_4_BIT
```

SEE ALSO

identify_register_banks(2)
split_register_bank(2)

create_route_guide

Creates a new route guide.

SYNTAX

```
collection create_route_guide
  -coordinate rectangle
  [-no_signal_layers layers]
  [-no_preroute_layers layers]
  [-standard_cell_region]
  [-zero_min_spacing]
  [-preferred_direction_only_layers layers]
  [-switch_preferred_direction_layers layers]
  [-repair_as_single_sbox]
  [-horizontal_track_utilization percentage]
  [-vertical_track_utilization percentage]
  [-track_utilization_layers layers]
  [-name guide_name]
  [-single_layer_routing layers]
  [-access_preference {access_preference_spec}]
```

Data Types

<i>rectangle</i>	list of points
<i>layers</i>	list of strings
<i>percentage</i>	integer
<i>guide_name</i>	string
<i>access_preference_spec</i>	list

ARGUMENTS

-coordinate *rectangle*

Specifies the coordinates of the lower-left and upper-right corners of the bounding box of the route guide. The coordinate unit is specified in the technology file.

This option is required.

-no_signal_layers *layers*

Specifies the layers that cannot contain signals. Specify the layers by using the layer names from the technology file.

-no_preroute_layers *layers*

This option is not supported in Design Compiler.

-standard_cell_region

Specifies that the type of route guide is `standard_cell_region`.

-zero_min_spacing

This option is not supported in Design Compiler.

-preferred_direction_only_layers *layers*

This option is not supported in Design Compiler.

-switch_preferred_direction_layers *layers*

This option is not supported in Design Compiler.

-repair_as_single_sbox

This option is not supported in Design Compiler.

-horizontal_track_utilization *percentage*

Specifies the horizontal track utilization rate in the area covered by the route guide.

You must specify an integer value between 0 and 100 for the *percentage* argument. If you do not specify this option or if you specify a value greater than 100, it is set to 100.

The following example specifies 90% horizontal track utilization route guide.

```
create_route_guide -horizontal_track_utilization 90
```

If an area has 10 horizontal tracks for routing, and Global Router puts 10 horizontal tracks here, with 90% horizontal utilization route guide, Global Router will now think the area is 1 track overflow.

-vertical_track_utilization *percentage*

Specifies the vertical track utilization rate in the area covered by the route guide.

You must specify an integer value between 0 and 100 for the *percentage* argument. If you do not specify this option or if you specify a value greater than 100, it is set to 100.

The following example specifies 90% vertical track utilization route guide.

```
create_route_guide -vertical_track_utilization 90
```

If an area has 10 vertical tracks for routing, and Global Router puts 10 vertical tracks here, with 90% vertical utilization route guide, Global Router will now think the area is 1 track overflow.

-track_utilization_layers *layers*

Specifies the layers on which track utilization is set by the **-horizontal_track_utilization** or **-vertical_track_utilization** option. Specify the layers by using the layer names from the technology file.

The direction of the track utilization is determined from the preferred direction of the specified layers. If the preferred direction of a specified layer is horizontal, the horizontal track utilization is set on that layer. If you specify more than one utilization in the same area, the router uses the minimum utilization.

-name *guide_name*

Specifies the name of the created route guide.

If you do not specify this option, the tool assigns a name to the route guide using the naming convention `RG#x`, where `x` is an integer value.

-single_layer_routing *layers*

This option is not supported in Design Compiler.

DESCRIPTION

This command creates a new route guide and returns a collection that contains the created route guide.

The tool uses the global snap settings to snap the bounding box of the route guide.

To remove a route guide, use the **remove_route_guide** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a route guide.

```
prompt> create_route_guide -coordinate {0.0 0.0 100.0 100.0} \  
-no_signal_layers M1 -no_preroute_layers M2 \  
-preferred_direction_only_layers M4
```

SEE ALSO

create_net_shape(2)
create_placement_blockage(2)
create_terminal(2)
create_user_shape(2)
create_via(2)
remove_route_guide(2)

create_rp_group

Creates relative placement groups.

SYNTAX

```
collection create_rp_group
  group_list
  [-design design_name]
  [-columns num_cols]
  [-rows num_rows]
  [-alignment bottom-left | bottom-pin | bottom-right]
  [-pin_align_name pin_name]
  [-utilization percentage]
  [-ignore]
  [-x_offset float]
  [-y_offset float]
  [-cts_option fixed_placement | size_only]
  [-route_opt_option fixed_placement | in_place_size_only]
  [-psynopt_option fixed_placement | size_only | all_optimization]
  [-move_effort low | medium | high]
  [-allow_keepout_over_tapcell false | true]
  [-allow_non_rp_cells]
  [-place_around_fixed_cells standard | physical_only | none | all]
  [-anchor_corner bottom-left | bottom-right | top-left | top-right | rp-location [-anchor_row integer] [-anchor_column integer]]
  [-placement_type bit_slice | compression | vertical_compression]
  [-group_orient default | N | FN | S | FS]
  [-cell_orient_opt]
  [-auto_blockage]
  [-disable_buffering]
  [-ignore_rows ignore_row_names_list]
  [-max_rp_width float]
  [-max_rp_height float]
```

Data Types

<i>group_list</i>	list
<i>design_name</i>	design
<i>num_cols</i>	integer
<i>num_rows</i>	integer
<i>pin_name</i>	string
<i>percentage</i>	float
<i>float</i>	percentage
<i>integer</i>	num_rows
<i>ignore_row_names_list</i>	list of strings

ARGUMENTS

group_list

Specifies the names for the relative placement (RP) groups you want to create in the specified design. Within a design, each group name must be unique. The tool generates a warning if you try to create a group that does not have a unique name.

The **create_rp_group** command creates one relative placement group for each item in the *group_list*.

-design *design_name*

Specifies the name of the design in which to create the new groups. If you do not specify a value for this option, the groups are created in the current design.

-columns *num_cols*

Specifies the number of columns into which objects can subsequently be placed, relative to each other.

If you do not specify a value for this option, the group is created with 1 column.

-rows *num_rows*

Specifies the number of rows into which objects can be subsequently placed, relative to each other.

If you do not specify a value for this option, the group is created with 1 row.

-alignment **bottom-left** | **bottom-pin** | **bottom-right**

Specifies the default alignment method to use when placing leaf cells and relative placement groups in the specified relative placement groups. If you do not specify this option, the tool uses bottom-left alignment.

You can override the default alignment method for a specific leaf cell or relative placement group when you add it to a relative placement group by using the **add_to_rp_group** command.

-pin_align *name pin_name*

Specifies the default alignment pin for the created relative placement groups. During placement, the tool uses the alignment pin's location to align the cells within a column.

You can override the alignment pin for a specific leaf cell when you add the cell to the group by using the **add_to_rp_group** command.

-utilization *percentage*

Specifies the utilization percentage to use for placement of this relative placement group. Utilization is represented as a floating-point number between 0.0 (representing 0%) and 1.0 (the default, representing 100%).

-ignore

Indicates that the tool is to ignore this relative placement group during placement and not treat it as a relative placement group. Instead, the tool places all of the group's parts individually, as it would normally do when there is no relative placement.

-x_offset *float*

Specifies a left coordinate (anchor) for the group, in microns, relative to the lower-left corner in the core area, for top-level relative placement group. It is ignored for a lower level relative placement group. Specifying only an x-offset allows the group to slide in the y-direction.

-y_offset *float*

Specifies a lower coordinate (anchor) for the group, in microns, relative to the lower-left corner in the core area, for top-level

relative placement group. It is ignored for a lower level relative placement group. Specifying only a y-offset allows the group to slide in the x-direction.

-cts_option fixed_placement | size_only

Specifies how to treat the cells in the relative placement group during clock tree synthesis and clock tree optimization. If you specify **fixed_placement**, the cells in the relative placement group are treated as fixed during **compile_clock_tree**, **optimize_clock_tree**, and **clock_opt** actions and cannot be sized or moved.

If you specify **size_only**, the cells in the relative placement group can only be sized. This option is applicable to the **compile_clock_tree**, **optimize_clock_tree**, and **clock_opt** commands.

The default of the option is **fixed_placement**.

This option applies only to top-level relative placement groups and is ignored for hierarchical relative placement groups. The top-level value is propagated to the hierarchical relative placement groups.

This option is not supported in Design Compiler topographical mode and will be ignored.

-route_opt_option fixed_placement | in_place_size_only

Specifies how to treat the cells in the relative placement group during the **route_opt** and **psynopt on_route** stages.

If you specify **fixed_placement**, the cells in the relative placement group are treated as fixed during the **route_opt** and **psynopt on_route** stages and cannot be sized or moved.

If you specify **in_place_size_only**, sizing changes are constrained for minimal engineering change order (ECO) placement changes. For example, a cell is sized to improve timing or design rule costs only if the newly sized cell can fit into any available space adjacent to the original cell location. The resulting transformation is verified to ensure it is legal.

The default is **fixed_placement**.

This option applies only to top-level relative placement groups and is ignored for hierarchical relative placement groups. The top-level value is propagated to the hierarchical relative placement groups.

This option is not supported in Design Compiler topographical mode and will be ignored.

-psynopt_option fixed_placement | size_only | all_optimization

Specifies how to treat the cells in the relative placement group during **psynopt** and **place_opt**.

If you specify **fixed_placement**, the cells in the relative placement group are treated as fixed during **psynopt** and cannot be sized or moved. This value is not recommended for **place_opt**.

If you specify **size_only**, the cells in the relative placement group can only be sized.

If you specify **all_optimization**, all the preroute optimization capabilities can be applied to cells in the relative placement group. This might result in losing some cells from the relative placement group as cells can be decomposed or deleted from the netlist.

The default is **size_only**.

This option applies only to top-level relative placement groups and is ignored for hierarchical relative placement groups. The top-level value is propagated to the hierarchical relative placement groups.

This option is not supported in Design Compiler topographical mode and will be ignored.

-move_effort low | medium | high

Controls the extent to which a relative placement group can be moved from its initial location to preserve the relative placement without violating relative placement constraints. The size of the region searched for placement of a relative placement group is progressively reduced as you reduce the effort from high to medium to low.

Coarse placement estimates an initial location for every top-level relative placement group. If those groups are placed at the same locations returned by coarse placement, the relative placement groups might need to be broken, which would violate the relative placement constraints for those groups.

Specifying high effort allows the higher movement of a relative placement group from its initial location in search of a location compared to low and medium efforts to maintain the relative placement constraints and potentially avoid breaking the relative placement group. The tradeoff for maintaining the relative placement group is reduced QoR because the group is moved from the location initially determined by coarse placement.

Specifying low effort indicates that the relative placement group is placed as close as possible to the initial location returned by coarse placement. The tradeoff for maintaining the location is a higher potential for breaking the relative placement groups and violating the relative placement constraints for some relative placement groups.

By default, the tool uses medium effort.

This option is not supported in Design Compiler topographical mode and will be ignored.

-allow_keepout_over_tapcell false | true

Specifies that the hard keepout in the relative placement group should be overlapped with tap cells if needed.

By default, the value is **false**.

This option is not supported in Design Compiler topographical mode and will be ignored.

-allow_non_rp_cells

Applying this option allows nonrelative placement cells to come in the unused area of relative placement groups during **place_opt** and **refine_placement**.

-place_around_fixed_cells standard | physical_only | none | all

Specifies how to treat fixed cells in the floorplan during legalization of relative placement groups.

If you specify **standard**, the relative placement groups are legalized around fixed standard cells and the legalization engine avoids fixed physical-only cells.

If you specify **physical_only**, the relative placement groups are legalized around fixed physical-only cells, and the legalization engine avoids fixed standard cells.

If you specify **all**, the relative placement groups are legalized around both fixed standard and fixed physical-only cells.

If you specify **none**, the legalization engine avoids both fixed standard and fixed physical-only cells during legalization of relative placement groups.

The default is **all**.

This option applies to top-level relative placement groups only and is ignored for hierarchical relative placement groups.

-anchor_corner bottom-left | bottom-right | top-left | top-right | rp-location

Specifies the corner for the anchor point that is set by using the **-x_offset** and **-y_offset** options.

If you specify **bottom-left** (the default), the anchor point corner is the lower-left corner of the relative placement group.

If you specify **bottom-right**, the anchor point corner is the lower-right corner of the relative placement group.

If you specify **top-left**, the anchor point corner is the upper-left corner of the relative placement group.

If you specify **top-right**, the anchor point corner is the upper-right corner of the relative placement group.

If you specify **rp-location**, the anchor point corner is the starting location of object at the row and column specified by the -

anchor_row and **-anchor_column** options. When you specify **rp-location**, the **-anchor_row** and **-anchor_col** options are required and the specified location must not be empty.

This option applies to top-level relative placement groups only and is ignored for hierarchical relative placement groups. When you specify **-anchor_corner bottom-right**, the relative placement group is anchored at the bottom-right corner at the specified x- and y-coordinates during legalization.

When many blockages are present, a slight deviation from the anchor point might occur.

-anchor_row *integer*

Specifies the row of the object for which the **-anchor_corner rp-location** option is specified.

Note that this option is applicable only to the **-anchor_corner rp-location** option and is not accepted for other values of the **-anchor_corner** option.

-anchor_column *integer*

Specifies the column of the object for which the **-anchor_corner rp-location** option is specified.

Note that this option is applicable only to the **-anchor_corner rp-location** and is not accepted for other values of the **-anchor_corner** option.

-placement_type bit_slice | compression | vertical_compression

Specifies how to place cells in relative placement groups. The default is **bit_slice**.

When you specify the **bit_slice** keyword, the next row starts after the tallest object in the row and the next column starts after the widest object in the column. Both the row and column alignments are preserved.

When you specify the **vertical_compression** keyword, columns are compressed such that no gap is between leaf cells, lower-level hierarchical relative placement groups, and keepouts in a column. Row alignment is not preserved.

When you specify the **compression** keyword, rows are compressed such that no gap is between leaf cells, lower-level hierarchical relative placement groups, and keepouts in a row. Column alignment is not preserved. If you set the **-alignment** option to **bottom-right** or **bottom-pin** and set this option to **compression**, the tool issues an error and does not create relative placement groups.

If both the **-utilization** and **-placement_type compression** options are specified, the utilization constraints are observed with gaps between leaf elements in a relative placement row.

The setting of this option does not propagate to child relative placement groups.

-group_orient default | N | FN | S | FS

Specifies the orientation of a relative placement group. If this value is not specified or is specified as **default** for any relative placement group, the tool chooses a suitable orientation for that relative placement group.

-cell_orient_opt

Applies automatic orientation on the cells in a relative placement group for optimizing wire length.

-auto_blockage

Disallows buffer insertion inside the area of relative placement groups during **psynopt** and **route_opt** activity.

-disable_buffering

Disallows buffer insertion on nets within relative placement groups during **psynopt** and **route_opt** activity.

-ignore_rows *ignore_row_names_list*

Specifies the site rows that must be ignored for the relative placement group cells.

To ignore site rows during relative placement, you must first define the **rp_ignore_row_name** attribute on the site rows to be ignored. The *ignore_row_names_list* argument is a list that includes the **rp_ignore_row_name** attribute value for each site row to be ignored.

This option applies only to top-level relative placement groups and is ignored for hierarchical relative placement groups.

-max_rp_width float

Specifies the maximum allowable width in microns for the top-level relative placement groups. It is ignored for the lower-level relative placement groups. If you specify this option, the tool tries to restrict the width of the top-level relative placement groups to the specified maximum value during **psynopt**, **place_opt** and **route_opt**.

-max_rp_height float

Specifies the maximum allowable height in microns for the top-level relative placement groups. It is ignored for the lower-level relative placement groups. If you specify this option, the tool tries to restrict the height of the top-level relative placement groups to the specified maximum value during **psynopt**, **place_opt** and **route_opt**.

DESCRIPTION

The **create_rp_group** command creates new relative placement groups. The new groups are created empty and contain no items. Add items to a group using the **add_to_rp_group** command. Remove groups using the **remove_rp_groups** command.

This command returns a collection containing the relative placement groups that are created. If no objects are created, the empty string is returned.

Relative placement groups provide for explicit user control of placement for a group of cells. A relative placement group is placed as a whole while maintaining the relative placement (or tiling) of the cells within the group as specified with the **add_to_rp_group** command. Relative placement is also known as "structured placement."

When placing a relative placement group, the placer automatically chooses the orientation for the group. For example, if the placer chooses the north orientation for a relative placement group, the first column of that relative placement group is placed at the left and subsequent columns are placed toward the right. If the placer chooses the flip-north orientation for a relative placement group, the first column of that relative placement group is placed at the right and subsequent columns are placed toward the left.

Relative placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **create_rp_group** to create a relative placement group:

```
prompt> create_rp_group grp_ripple -design ripple -rows 8
{ripple::grp_ripple}
```

```
prompt> get_rp_groups grp_ripple
{ripple::grp_ripple}
```

```
prompt> add_to_rp_group *ripple -leaf carry_in_1 -row 2
{ripple::grp_ripple}

prompt> define_user_attribute -type string \
-class site_row rp_ignore_row_name
prompt> set_attribute [get_site_rows STD_ROW_112] \
rp_ignore_row_name clock
prompt> create_rp_group rp -design d -ignore_rows { clock }
```

SEE ALSO

```
add_to_rp_group(2)
all_rp_groups(2)
remove_rp_groups(2)
report_rp_group_options(2)
set_rp_group_options(2)
write_rp_groups(2)
```

create_scenario

Creates a scenario in memory.

SYNTAX

```
status create_scenario  
      scenario_name
```

Data Types

```
scenario_name  string
```

ARGUMENTS

scenario_name

Specifies the name of the scenario created. A *scenario_name* must be unique within the current session.

DESCRIPTION

This command creates a new scenario in memory. The newly-created scenario has no scenario-specific constraints.

The creation and usage of scenarios is available with Design Compiler Graphical.

Multicorner-Multimode Support

This command creates a new scenario for use with multicorner-multimode designs.

EXAMPLES

The following example uses the **create_scenario** command to create a scenario:

```
prompt> create_scenario MODE1  
prompt> all_scenarios
```

SEE ALSO

- `all_scenarios(2)`
- `all_active_scenarios(2)`
- `create_scenario(2)`
- `current_scenario(2)`
- `remove_scenario(2)`
- `set_active_scenarios(2)`

create_site_row

Creates a row of sites.

SYNTAX

```
collection create_site_row
  -coordinate {X Y}
  [-name row_name]
  -kind site_type
  -space space
  -count site_count
  [-orient orientation]
  [-dir direction]
```

Data Types

<i>X</i>	float
<i>Y</i>	float
<i>row_name</i>	string
<i>site_type</i>	string
<i>space</i>	string
<i>site_count</i>	integer
<i>direction</i>	string

ARGUMENTS

-coordinate {*X Y*}

Specifies the lower-left corner of the row, regardless of the orientation. The values are specified in microns relative to the chip origin.

-name *row_name*

Specifies the name of the site row, such as *ROW1*, *ROW2*, and so forth. If a row name is not specified, the tool names the newly-created site row.

-kind *site_type*

Specifies the type of site being defined.

-space *space*

Specifies the space for each site, from the lower-left corner of the site to the lower-left corner of the next site. The value is specified in microns.

-count *site_count*

Specifies the number of sites in the row.

-orient *orientation*

Specifies the orientation of sites. The following values are allowed:

0
90
180
270
0-mirror
90-mirror
180-mirror
270-mirror

Alternatively, you can use the following values:

N
W
S
E
FN
FW
FS
FE

The default value is either **0** or **N**.

-dir *direction*

Specifies the direction of the row. The following values are allowed:

h
v
horizontal
vertical
VERTICAL
HORIZONTAL
V
H

The default value is **horizontal**.

DESCRIPTION

The **create_site_row** command creates a row of sites at the specified location.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command creates a horizontal row of 3 sites spaced by 5 units, and originating at {10,10}:

```
prompt> create_site_row -coordinate {10,10} -kind CORE -space 5 -count 3  
{SITE_ROW#12345}
```

SEE ALSO

`extract_physical_constraints(2)`
`report_physical_constraints(2)`
`write_physical_constraints(2)`

create_supply_net

Creates a supply net for the specified power domain. The supply net is created in the logic hierarchy at the same scope as the specified power domain.

SYNTAX

```
string create_supply_net  
    supply_net_name  
    [-domain domain_name]  
    [-reuse]  
    [-resolve unresolved | parallel | one_hot | parallel_one_hot | resolution_function_name]
```

Data Types

```
supply_net_name    string  
domain_name       string
```

ARGUMENTS

supply_net_name

Specifies the name of the supply net to be created. The name should be a simple (nonhierarchical) name.

In the scope of specified power domain, if there is a supply net, logical hierarchical net, or logical port with the same name, the supply net cannot be created. An exception to this rule is when you use the **-reuse** option. In this case the net name must be same as an existing supply net in the same scope; name; check is not performed.

This argument is required.

-domain *domain_name*

Specifies the power domain for which this supply net is defined. The power domain must exist.

This argument is optional. If -domain is not specified, the supply net is available for all power domains at the current scope and below.

-reuse

Reuses an existing supply net instead of creating a new one. One supply net can be associated with several power domains. If this option is used, the specified supply net must already exist.

-resolve unresolved | parallel | one_hot | parallel_one_hot | resolution_function_name

Specifies how the state and voltage of the supply net are resolved when the supply net is driven by a power switch or multiple power switches. Specifying *unresolved* means that this supply net can have only a single driver. Specifying *parallel* means that this supply net can be driven by multiple power switches. When the supply net is connected to the output of multiple power

switches, any number of outputs might be ON at the same time, if the voltage value is same. Specifying *one_hot* means that this supply net can be driven by multiple power switches. A supply net with *one_hot* resolution has a deterministic state only when no more than one source drives the net at any particular point in time. Specifying *parallel_one_hot* means that this supply net can be driven by multiple power switches. The *parallel_one_hot* resolution allows resolution of a supply net that has multiple root supply drivers where each driver might have more than one path through supply sources to the supply net. Each unique root supply driver is identified and *one_hot* resolution is applied to the drivers. *Parallel* resolution is then applied to each supply source connecting the *one_hot* root supply driver to the supply net. Resolution semantics may also be specified by a user-defined resolution function. When a user-defined resolution function is specified, there are no restrictions on the number of input sources or the number of root supply drivers involved, and the function is responsible for defining any restrictions on the values of inputs as well as the algorithm for determining the output result.

The default is **unresolved**.

DESCRIPTION

The **create_supply_net** command enables you to create a supply net defined in the specified power domain. A supply net presents the intention of the power supply in the power domains. A supply net connects supply ports and, or pins.

Each power domain has a primary power supply net and a primary ground supply net, and could have several other supply nets. If the command succeeds, a supply net is created in the scope of the power domain. The new supply net is neither a primary power net nor a primary ground net of the power domain. Use the **set_domain_supply_net** command to set the supply net as the primary power or ground net.

On success this command returns the full name of the supply net (from the current scope). On failure, it returns a null string.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a supply net named A_VDD in the power domain PD1, and re-creates supply net A_DD in the power domain PD2 using the **-reuse** option:

```
prompt> create_power_domain PD1 -element INST1
PD1

prompt> create_power_domain PD2 -element INST2
PD2

prompt> create_supply_net A_VDD -domain PD1
A_VDD

prompt> create_supply_net A_VDD -domain PD2 -reuse
A_VDD
```

SEE ALSO

`report_supply_net(2)`

create_supply_port

Creates a supply port in the specified power domain. If a power domain is not specified, creates the port in the current scope.

SYNTAX

```
string create_supply_port  
    supply_port_name  
    [-domain domain_name]  
    [-direction in | out]
```

Data Types

```
supply_port_name  string  
domain_name       string
```

ARGUMENTS

supply_port_name

This is a required argument.

Specifies the name of the supply port to be created. The name should be a simple (non-hierarchical) name. This argument can be used with the **-domain** option to avoid hierarchical names.

If a supply port, logical port, or logical hierarchical net with the same name exists in the scope of the power domain, the supply port cannot be created.

-domain *domain_name*

Specifies the power domain where this port defines a supply net connection point. If the specified power domain does not exist in the current scope, the command fails.

-direction *in | out*

Defines how the state information is propagated through the supply network when connected to the port. If the port is an input port, the state information of the external supply net connected to the port is propagated into the domain. Similarly, for an output port, the state information of the internal supply net connected to the port is propagated outside the domain.

The default value of this option is **in**.

DESCRIPTION

The *create_supply_port* command creates a supply port at the scope of the specified power domain or at the current scope. A supply port provides a connection point for the supply net.

On success, the command returns the full name of the supply port (from the current scope). Otherwise, returns a null string.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates the supply port *VN1* at the scope of power domain *PD1*:

```
prompt> create_power_domain PD1 -element INST1
PD1
```

```
prompt> create_supply_port VN1 -domain PD1
VN1
```

SEE ALSO

`report_supply_port(2)`

create_supply_set

Creates a set of supplies that can be used to define the power network. A supply set is created in the current logic hierarchy.

SYNTAX

```
string create_supply_set  
  [-function {function_name supply_net_name}]  
  [-update]  
  supply_set_name
```

Data Types

```
function_name    string  
supply_net_name  string  
supply_set_name  string
```

ARGUMENTS

-function {function_name supply_net_name}

Specifies the functionality to which an actual supply net should be associated.

This option takes a list of two parameters as input. The first parameter is the name of the function, which is either **power** or **ground**. The second parameter is the name of the supply net that is in the same hierarchical scope as the supply set.

You can specify the **-function** option multiple times in a single run of the **create_supply_set** command; once to define the power supply net and once to define the ground supply net.

This is a required argument when the **-update** option is used.

-update

Instructs the tool to associate an actual supply net to the **power** and **ground** functions of an existing supply set. It is an error if the supply set specified does not exist. Only one update is allowed for each function of the supply set. If your update is not successful, the tool issues an error message. You can continue to update the function, until your update is successful.

This is an optional argument. When the **-update** argument is used, the **-function** argument must also be used.

supply_set_name

Specifies the name of the supply set to be created. The name should be a simple (non-hierarchical) name.

If a supply set with the same name exists in the current scope, the supply set is not created. An exception to this rule is when you use the **-update** option. In this case the supply set name must be same as an existing supply set in the same scope.

This is a required argument.

DESCRIPTION

The **create_supply_set** command creates a supply set in the current scope. A supply set eliminates the requirement of a supply net and supply ports in the design, in the front-end tool. However, actual supply nets must be associated with the supply sets, later in the flow, before running physical synthesis.

A supply set supports the use of the following functions in the design:

- **power**
- **ground**

These functions, when unassociated with any supply net, can be used as supply nets for any purpose. They can be accessed by referencing them through the name of the supply set. To access the **power** function, you must specify *supply_set_name.power*. To access the **ground** function, you must specify *supply_set_name.ground*.

The functions of a supply set can be used just as any other supply net in the UPF design, with no domain restrictions. However, domain restrictions are applicable when you run the command with the **-update** option. This means that after you run the command with the **-update** option, the association of a supply net with a supply set is checked in the domains where the supply set is used.

When you run the command with the **-update** option on an existing supply set, the command replaces the supply set functions with actual supply nets. After doing an update, the actual supply net can be referenced through the supply set or by its actual name.

EXAMPLES

The following example creates a supply set named `primary_sset` in the current logical hierarchy:

```
prompt> create_supply_set primary_sset
primary_sset
```

The following example illustrates how a supply set function can be used:

```
prompt> create_supply_set primary_sset
primary_sset
```

```
prompt> create_power_domain PD_TOP
PD_TOP
```

```
prompt> set_domain_supply_net PD_TOP A \
  -primary_power_net primary_sset.power \
  -primary_ground_net primary_sset.ground \
1
```

```
prompt> set_retention ret_cstr -domain PD_TOP \
  -retention_power_net primary_sset.power \
  -retention_ground_net primary_sset.ground
1
```

```
prompt> set_isolation iso_cstr -domain PD_TOP \
  -isolation_power_net primary_sset.power \
  -isolation_ground_net primary_sset.ground \
```

1

The following example illustrates how to associate a supply set with actual supply nets.

```
prompt> create_supply_set primary_sset  
primary_sset  
  
prompt> create_power_domain PD_TOP  
PD_TOP  
  
prompt> set_domain_supply_net PD_TOP \  
-primary_power_net primary_sset.power \  
-primary_ground_net primary_sset.ground  
1  
  
prompt> set_retention ret_cstr -domain PD_TOP \  
-retention_power_net primary_sset.power \  
-retention_ground_net primary_sset.ground  
1  
  
prompt> set_isolation iso_cstr -domain PD_TOP \  
-isolation_power_net primary_sset.power \  
-isolation_ground_net primary_sset.ground \  
1  
  
prompt> create_supply_net TOP_VDD -domain PD_TOP  
TOP_VDD  
  
prompt> create_supply_net TOP_VSS -domain PD_TOP  
TOP_VSS  
  
prompt> create_supply_set primary_sset \  
-function {power TOP_VDD} \  
-function {ground TOP_VSS} \  
-update  
1
```

SEE ALSO

`create_power_domain(2)`
`create_supply_net(2)`
`set_domain_supply_net(2)`
`set_isolation(2)`
`set_retention(2)`

create_terminal

Creates a new terminal for a logical port.

SYNTAX

```
collection create_terminal
  -bbox rect | -bounding_box rect
  -layer layer
  [-mask_constraint constraint]
  -port port_name
  [-direction {pin_directions}]
  [-name terminal_name]
```

Data Types

<i>rect</i>	list of two points
<i>layer</i>	collection
<i>constraint</i>	string
<i>port_name</i>	string
<i>terminal_name</i>	string

ARGUMENTS

-bounding_box *rect* | -bbox *rect*

Specifies the bounding box of the terminal in the format {*x1 y1 x2 y2*}. These options are equivalent; you can specify only one of them. In addition, these options are mutually exclusive; you must specify one of them.

-layer *layer*

Specifies the layer of the terminal.

This option is required.

-mask_constraint *constraint*

Specifies the mask constraint for the terminal.

The valid mask constraints are **any_mask**, **mask1_soft**, **mask1_hard**, **mask2_soft**, **mask2_hard**, **same_mask**, **mask3_soft**, and **mask3_hard**.

-port *port_name*

Specifies the name of the associated logical port of the terminal.

This option is required.

-direction {*pin_directions*}

Specifies a list of allowable access directions for the terminal.

You can specify one or more of the following values: **left**, **right**, **up**, or **down**.

If you do not specify this option, no access direction is assumed.

-name *terminal_name*

Specifies the name of the created terminal.

If you do not specify this option, the tool automatically generates a terminal name from the port name.

In most cases, you should not specify this option and allow the tool to automatically generate the terminal name, because some operations require that the terminal name conforms to the terminal naming conventions. For information about the terminal naming conventions, see the DESCRIPTION section.

If the specified terminal name does not match the terminal naming conventions, the tool issues a warning.

DESCRIPTION

The **create_terminal** command creates a new terminal for a logical port and returns a collection containing the created terminal.

By convention, the terminal name is the same as the name of the associated port; however, this is not a strict requirement. For electrically equivalent (EEQ) pins, where there are multiple terminals associated with the same port, the first terminal has the same name as the port and subsequent terminal names have the format *port_name number* where *port_name* is the port name and *number* is a unique number which, when automatically generated, starts from one and increments for each new terminal on that port.

Snapping of the bounding box is done automatically using the global snap settings.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a terminal for the A1 port on the M3 layer:

```
prompt> create_terminal -bbox {0 0 10 10} -port A1 -layer M3
```

SEE ALSO

create_net_shape(2)
create_placement_blockage(2)
create_route_guide(2)
create_user_shape(2)

create_via(2)

create_test_protocol

Creates a test protocol based on user specifications.

SYNTAX

```
status create_test_protocol
  [-infer_asynch]
  [-infer_clock]
  [-capture_procedure single_clock | multi_clock]
```

ARGUMENTS

-infer_asynch

Infers asynchronous set and reset signals in the design.

-infer_clock

Infers test clocks in the design.

-capture_procedure single_clock | multi_clock

Specifies the capture procedure type. The **multi_clock** type creates a protocol file that uses generic capture procedures for all capture clocks. The **single_clock** type creates a protocol file that uses the legacy 3-vector capture procedures for all capture clocks.

The default is **multi_clock**.

DESCRIPTION

The **create_test_protocol** command creates a test protocol for the current design based on user specifications issued prior to running this command. The specifications are made using commands such as **set_dft_signal**.

This command removes any protocol that is present in memory due to a previous execution of **create_test_protocol**. However, if the protocol is present in memory due to a previous execution of **read_test_protocol**, it issues a warning and does not create a new test protocol.

This command checks whether the user-specified values are consistent with each other. If they are not, it issues an error and does not generate a protocol.

If **-infer_asynch** is specified, **create_test_protocol** infers asynchronous set and reset signals in the design, and places them at off state during scan shifting.

If **-infer_clock** is specified, **create_test_protocol** infers test clock pins from the design, and pulses them during scan shifting. The timing of the test clock is based on the **test_default_period**, **test_default_delay**, **test_default_strobe**, and **test_default_strobe_width** variables.

Both **-infer_asynch** and **-infer_clock** take previous user specifications into account.

When the default **-capture_procedure_multi_clock** is specified, the protocol file contains four capture procedures: multiclock_capture, allclock_capture, allclock_launch, and allclock_launch_capture. This single protocol file can be used for ATPG stuck-at, transition delay, and path delay testing.

The **create_test_protocol** command automatically generates a master_observe procedure for LSSD designs in STIL format. However, to use this feature, you must set the scan style to LSSD design by using the **set_scan_configuration -style** command.

The **create_test_protocol** command should be executed before running the **dft_drc** command because design rule checking requires a test protocol.

If the interface of a design changes, for example, a port is created or removed, after a test protocol is created, there is no need to rerun **create_test_protocol** because the interface change is automatically taken into account in the protocol.

However, if you change any test specification of the design, for example, if you use **set_dft_signal** to specify a test clock, the protocol present in the memory is deleted. In this case, you need to rerun **create_test_protocol** to create a test protocol. The commands that change the test specifications of the existing design include **set_dft_signal** and **set_scan_path**.

The **insert_dft** command automatically updates the protocol after inserting scan circuitry into the design, and the **dft_drc** command can be executed afterward without rerunning **create_test_protocol**.

EXAMPLES

The following example creates a test protocol in memory for the current design:

```
prompt> create_test_protocol
```

The following example creates a test protocol in memory for the current design, inferring both asynchronous set and reset signals, as well as test clocks:

```
prompt> create_test_protocol -infer_clock -infer_asynch
```

SEE ALSO

- current_design(2)
- read_test_protocol(2)
- remove_test_protocol(2)
- set_dft_signal(2)
- set_scan_configuration(2)
- set_scan_path(2)
- write_test_protocol(2)

create_track

Creates tracks for a routing layer or poly layer.

SYNTAX

```
int create_track  
  -layer layer  
  [-space track_pitch]  
  [-count number_of_tracks]  
  [-coord start_x_or_y]  
  [-dir X | Y]  
  [-bounding_box track_boundary_box]  
  [-mask_constraint constraint]
```

Data Types

<i>layer</i>	string
<i>track_pitch</i>	float
<i>number_of_tracks</i>	integer
<i>start_x_or_y</i>	float
<i>track_boundary_box</i>	rectangle

ARGUMENTS

-layer *layer*

Specifies the layer to use the tracks. You can specify the layer name, the layer number, or a collection containing one layer object.

-space *track_pitch*

Specifies the pitch between tracks. The pitch is the center-to-center distance between two routing wires in adjacent tracks. By default, this distance is the same as the routing pitch from the unit tile in the Milkyway reference library. The pitch unit size is specified in technology file, typically microns.

-count *number_of_tracks*

Specifies how many tracks to create. By default, the tracks use the entire die area. If you do not specify this option, the value is automatically calculated from the available area of the die or bounding box.

-coord *start_x_or_y*

Specifies the x-coordinate or y-coordinate of the first track, depending on whether the tracks are vertical or horizontal, as determined by the **-dir** option. The unit size for the coordinate is specified in the technology file, typically microns. By default, the coordinate is the left or bottom coordinate of the bounding box, or one-half the pitch inside of the die area.

-dir X | Y

Specifies the direction in which the tracks are placed, either **X** (horizontal) or **Y** (vertical). By default, the direction is the preferred routing direction of the layer, which is specified by the unit tile in the Milkyway reference library.

-bounding_box track_boundary_box

Specifies the lower-left and upper-right coordinates of the bounding box that encloses all the tracks, in the form of `{{x1 y1} {x2 y2}}`. The unit size for the coordinates is specified in the technology file, typically microns. By default, there is no bounding box and the tracks cover the available die area.

-mask_constraint constraint

Specifies the mask constraint for the terminal.

The valid mask_constraints are **any_mask**, **mask1_soft**, **mask1_hard**, **mask2_soft**, **mask2_hard**, **same_mask**, **mask3_soft**, **mask3_hard**.

DESCRIPTION

This command creates a group of tracks on the floorplan so the router can use them to perform detail routing, or the **insert_metal_filler** command can use them to fill the poly layer at the chip finishing stage.

You must specify either a routing layer or poly layer for the tracks. The option of creating tracks for a poly layer is intended for fill purposes only, not routing.

By default, the command creates tracks the preferred routing direction and fills the available die area, starting from one-half the pitch from the die corner. You can optionally specify the track direction (X or Y), the coordinate of the first track, the number of tracks to create, or a bounding box to be filled with tracks.

The tracks created by this command are part of the design database. They are saved in the Milkyway database when you use the **save_mw_cel** command and in DEF format when you use the **write_def** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates routing tracks for a routing layer named "m3" on the floorplan and reports the newly created routing tracks.

```
prompt> create_track -layer m3
Warning: Space is not specified. Using pitch. (MWUI-124)
Warning: Direction is not specified. Using the layer preferred direction.
(MWUI-125)
Warning: Coordinate value is not specified. Defaulting to the left(bottom)
coordinate of the track's bounding box. (MWUI-126)
Warning: Count value is not specified. Covering the bounding box of track,
depending on its space value. (MWUI-127)
1
prompt> report_track -layer m3
```

```

...
Layer      Direction  Start  Tracks  Pitch  Attr
-----
Attributes :
  usr : User defined
  def : DEF defined

m3         Y      0.280   3696   0.560  usr
1

```

The following example creates reserved tracks for a routing layer named "m2" with width of 0.06, using a nondefault routing rule, and reports the newly created tracks.

```

prompt> create_track -layer m2 -dir Y -coord 0.438 -space 0.57 \
  -width 0.06 -bounding_box [list "0.000 0.12" "190.260 341.88"] \
  -reserved_for_width true
Warning: Count value is not specified. Covering the bounding box of track,
depending on its space value. (MWUI-127)
1
prompt> report_track -layer m2
...
Layer      Direction  Start  Tracks  Pitch  Attr
-----
Attributes :
  usr : User defined
  def : DEF defined

m2         Y      0.438    790   0.570  usr,width=0.060,reserved_for_width
1

```

SEE ALSO

```

get_tracks(2)
remove_track(2)
report_track(2)

```

create_user_shape

Creates a new user shape. A user shape is a metal shape that is not associated with a net.

SYNTAX

```
collection create_user_shape
  [-type wire | path | trap | rect | poly]
  -origin point
  | -points list_of_points
  | -bbox rect
  | -boundary boundary
  [-length length]
  [-width width]
  [-path_type square | round | extend_half_width | octagon]
  -layer layer
  [-mask_constraint constraint]
  [-vertical]
  [-route_type route_type]
  [-datatype int]
  [-avoid_short_segment]
```

Data Types

<i>point</i>	point
<i>list_of_points</i>	list of points
<i>rect</i>	rectangle
<i>boundary</i>	polygon
<i>length</i>	float
<i>width</i>	float
<i>layer</i>	collection
<i>constraint</i>	string
<i>route_type</i>	string
<i>int</i>	integer

ARGUMENTS

-type wire | path | trap | rect | poly

Specifies the type of the user shape. Valid values are **wire**, **path**, **trap** (trapezoid), **rect** (rectangle), and **poly** (polygon).

The default is **poly**.

-origin *point*

Specifies the origin of the user shape for wires.

When you specify this option, you must also specify the **-length** and **-width** options.

The **-origin**, **-points**, **-bbox**, and **-boundary** options are mutually exclusive. You must specify one of these options.

-points *list_of_points*

Specifies the points of the user shape for paths.

When you specify this option, you must also specify the **-width** option.

The **-origin**, **-points**, **-bbox**, and **-boundary** options are mutually exclusive. You must specify one of these options.

-bbox *rect*

Specifies the bounding box of the user shape. You must specify the lower-left and upper-right corners of the bounding box by using the following syntax: `{{llx lly} {urx ury}}`.

The **-origin**, **-points**, **-bbox**, and **-boundary** options are mutually exclusive. You must specify one of these options.

-boundary *boundary*

Specifies the boundary of the user shape. You must specify at least three nonoverlapping points.

The **-origin**, **-points**, **-bbox**, and **-boundary** options are mutually exclusive. You must specify one of these options.

-length *length*

Specifies the length of the user shape for wires in user units.

This option is required when you specify the **-origin** option and is ignored otherwise.

-width *width*

Specifies the width of the user shape for wires or paths in user units.

This option is required when you specify the **-origin** or **-points** option and is ignored otherwise.

-path_type *square* | *round* | *extend_half_width* | *octagon*

Specifies the alignment type of a wire or path end.

The default is **square**.

You can specify one of the following values:

- **square** (square, no extension)
- **round** (round, half-width extension)
- **extend_half_width** (square, half-width extension)
- **octagon** (octagon, half-width extension)

This option can be used only for wire or path shapes.

-layer *layer*

Specifies the layer for the user shape. You can specify the layer by using the layer name from the technology file or by using the **get_layers** command.

This is a required option.

-mask_constraint *constraint*

Specifies the mask constraint for the user shape.

The valid mask constraints are **any_mask**, **mask1_soft**, **mask1_hard**, **mask2_soft**, **mask2_hard**, **same_mask**, **mask3_soft**, and **mask3_hard**.

-vertical

Indicates that the user shape is vertical.

By default, the user shape is horizontal.

-route_type *route_type*

Specifies the route type of the user shape.

By default, the route type is **signal_route**.

You can specify one of the following values:

- **user_enter** (user entered)
- **signal_route** or **signal_route_detail** (detail routing)
- **signal_route_global** (global routing)
- **pg_ring** (power or ground ring)
- **pg_strap** (power or ground strap)
- **pg_macro_io_pin_conn** (power or ground net that connects to the power or ground pin of an I/O pad cell or a macro cell)
- **pg_std_cell_pin_conn** (power or ground net that connects to the power or ground pin of a standard cell)
- **clk_ring** (clock ring)
- **clk_strap** (clock strap)
- **clk_zero_skew_route** (clock zero-skew route)
- **bus** (bus)
- **shield** or **shield_fixed** (fixed shield)
- **shield_dynamic** (dynamic shield)
- **fill_track** or **clk_fill_track** (fill track)

This option can be used only for wire, path, or rectangle shapes.

-datatype *int*

Specifies the data type number in the range 0-255.

By default, the data type is 0.

-avoid_short_segment

Avoids creating segments shorter than a half width for paths.

DESCRIPTION

This command creates a shape object that is not attached to a net and returns a collection that contains the created user shape.

The valid shape types are wire (horizontal or vertical), path, trapezoid, rectangle, and polygon.

Note: For wires, the shape must be symmetrical about the center line. If you specify the width as an odd value, either explicitly by using the **-width** option or implicitly by using the **-bbox** option, the tool rounds the width down to an even value.

For example, the following command specifies a width of 9995 database units. If the technology file defines the lengthPrecision as 1000, the tool rounds this down to 9994, resulting in a bounding box with the coordinates (100.000, 100.000) (**109.994**, 169.895).

```
prompt> create_user_shape -type wire -route_type pg_strap \  
-layer M6 -bbox { {100.000 100.000} {109.995 169.895} } -vertical
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a wire user shape.

```
prompt> create_user_shape -type wire \  
-origin {0 0} -length 10 -width 2 -layer M1
```

SEE ALSO

- create_net_shape(2)
- create_placement_blockage(2)
- create_route_guide(2)
- create_via(2)
- create_terminal(2)

create_via

Creates a new via at a specified location.

SYNTAX

```
collection create_via
  -at point
  [-name name]
  -master via_master | -auto
  [-net net_name | -no_net]
  [-allow_multiple]
  [-route_type route_type]
  [-orient orientation]
  [-type via | via_array]
  [-row num_rows]
  [-col num_columns]
  [-x_pitch x_distance]
  [-y_pitch y_distance]
```

Data Types

<i>point</i>	point
<i>name</i>	string
<i>via_master</i>	collection
<i>net_name</i>	string
<i>route_type</i>	string
<i>orientation</i>	string
<i>num_rows</i>	integer
<i>num_columns</i>	integer
<i>x_distance</i>	float
<i>y_distance</i>	float

ARGUMENTS

-at *point*

Specifies the location of the center of the created via or via array in the format {*x y*}. The units for the coordinates are microns.

This argument is required.

-name *name*

Specifies the name of the created via or via array.

If you do not specify this option, the tool assigns a name using the naming convention *VIA#n*.

-master *via_master*

Specifies the via master. You can specify the via master by using its name or a collection of one via master.

To determine the available via masters, use the **get_via_masters** command.

When you use this option, you must also use either the **-net** or **-no_net** option.

-auto

This option is not supported in Design Compiler topographical mode.

-net *net_name*

Specifies the name of the net to connect to the via.

This option and the **-no_net** option are mutually exclusive; you must specify one of these options when you use the **-via_master** option.

-no_net

Specifies that no net should be connected to the via.

This option and the **-net** option are mutually exclusive; you must specify one of these options when you use the **-via_master** option.

-allow_multiple

This option is not supported in Design Compiler topographical mode.

-route_type *route_type*

Specifies the route type of the via.

Specify one of the following values:

user_enter	User entered
signal_route, signal_route_detail	Detail routing
signal_route_global	Global routing
pg_ring	Power or ground (PG) ring
pg_strap	PG strap
pg_macro_io_pin_conn	PG net that connects to the PG pin of an I/O pad cell or a macro cell
pg_std_cell_pin_conn	PG net that connects to the PG pin of a standard cell
clk_ring	Clock ring
clk_strap	Clock strap
clk_zero_skew_route	Clock zero skew route
bus	Bus
shield, shield_fixed	Fixed shield
shield_dynamic	Dynamic shield
fill_track, clk_fill_track	Fill track

By default, the route type is **signal_route**.

-orient *orientation*

Specifies the orientation of the via or via array using either DEF notation or the tool notation.

Specify one of the following values (values listed in the same bullet are synonymous):

- N, NE, 0

- W , WN, 90
- S , SW, 180
- E , ES, 270
- FN, NW, 0-mirror
- FS, SE, 180-mirror
- FE, EN, 270-mirror
- FW, WS, 90-mirror

By default, the orientation is **N**.

-type via | via_array

Specifies the type of via to create.

By default, a single-cut via is created unless you specify another command option that is specific to via arrays.

-row num_rows

Specifies the number of via array rows.

-col num_columns

Specifies the number of via array columns.

-x_pitch x_distance

Specifies the x-distance in microns between cut centers in the via array.

-y_pitch y_distance

Specifies the y-distance in microns between cut centers in the via array.

This pitch values must be greater than or equal to the sum of the via cut dimension and the minimum cut spacing specified in the technology file for the specified via master. If the value is less than this threshold, a warning is issued and the value is adjusted to the minimum allowable value.

DESCRIPTION

The **create_via** command creates a new via or via array and returns a collection that contains the created object.

Snapping is performed automatically by using the global snap settings.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates an instance of a VIA12 via master at location (100,100) without connecting the via to any net, and identifies the route type as "clock ring":

```
prompt> create_via -at {100 100} -route_type clk_ring \  
        -master VIA12 -no_net  
        {VIA#89088}
```

SEE ALSO

- create_net_shape(2)
- create_placement_blockage(2)
- create_route_guide(2)
- create_terminal(2)
- create_user_shape(2)

create_via_master

Creates a design via master using parameters or a geometry list.

SYNTAX

```
status create_via_master
-name via_master_name
-cut_layer_name cut_layer_name
-lower_layer_name low_layer_name
-upper_layer_name up_layer_name
[-rectangles {{layer_name {llx lly urx ury} ...}}]
[-cut_width cut_width]
[-cut_height cut_height]
[-lower_layer_enc_width low_enc_width]
[-lower_layer_enc_height lo_enc_height]
[-upper_layer_enc_width up_enc_width]
[-upper_layer_enc_height up_enc_height]
[-min_cut_spacing min_cut_spacing]
[-quiet]
```

Data Types

```
via_master_name   string
cut_layer_name    string
low_layer_name    string
up_layer_name     string
layer_name        string
cut_width         float
cut_height        float
low_enc_width     float
lo_enc_height     float
up_enc_width      float
up_enc_height     float
min_cut_spacing  float
```

ARGUMENTS

-name *via_master_name*

Specifies the name of the created via master.

-cut_layer_name *cut_layer_name*

Specifies the cut layer for the via master.

-lower_layer_name *low_layer_name*

Specifies the lower metal layer for the via master.

-upper_layer_name *up_layer_name*

Specifies the upper metal layer for the via master.

-rectangles {{*layer_name* {*llx lly urx ury*}} ...}

Specifies the list of rectangles that describe the geometry of the via master.

If you use this option, the command ignores the following options: **-cut_width**, **-cut_height**, **-lower_layer_enc_width**, **-lower_layer_enc_height**, **-upper_layer_enc_width**, **-upper_layer_enc_height**, and **-min_cut_spacing**.

-cut_width *cut_width*

Specifies the cut width in microns of the via master.

You cannot use this option if you use the **-rectangles** option.

-cut_height *cut_height*

Specifies the cut height in microns of the via master.

You cannot use this option if you use the **-rectangles** option.

-lower_layer_enc_width *low_enc_width*

Specifies the lower metal layer enclosure width in microns of the via master. The value can be either positive or negative. The negative range is down to -25% of cut size.

You cannot use this option if you use the **-rectangles** option.

-lower_layer_enc_height *lo_enc_height*

Specifies the lower metal layer enclosure height in microns of the via master. The value can be either positive or negative. The negative range is down to -25% of cut size.

You cannot use this option if you use the **-rectangles** option.

-upper_layer_enc_width *up_enc_width*

Specifies the upper metal layer enclosure width in microns of the via master. The value can be either positive or negative. The negative range is down to -25% of cut size.

You cannot use this option if you use the **-rectangles** option.

-upper_layer_enc_height *up_enc_height*

Specifies the upper metal layer enclosure height in microns of the via master. The value can be either positive or negative. The negative range is down to -25% of cut size.

You cannot use this option if you use the **-rectangles** option.

-min_cut_spacing *min_cut_spacing*

Specifies the minimum spacing in microns between cuts when the via master is used to form an array.

You cannot use this option if you use the **-rectangles** option.

-quiet

Suppresses error and warning messages.

DESCRIPTION

This command creates a new via master. You use this command when the technology file does not already have a via master that matches the required geometry. The working design in which you want to use the via must be open before you create the new via master.

You can create the new via master either by specifying its geometry as a list of rectangles or by specifying the width, height, lower-layer enclosure, upper-layer enclosure, and minimum cut-spacing characteristics of the via, similar to a technology file definition.

To create a single-cut, symmetrical via master, use the height, width, enclosure, and minimum spacing options. To create a multi-cut via master such as an H-shape, use the **-rectangles** option.

When you save the current working cell, the via master is saved with the current cell. You can use the new via master any number of times.

If you specify a geometry list using the **-rectangles** option and you also specify parameters such as width, height, enclosure, and minimum spacing, the tool ignores the parameter settings and issues a warning.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command creates a single-cut via master using parameters:

```
prompt> create_via_master \  
-name VIA1 \  
-cut_layer_name CUT \  
-lower_layer_name METAL1 \  
-upper_layer_name METAL2 \  
-cut_width 0.22 \  
-cut_height 0.22 \  
-lower_layer_enc_width 0.1 \  
-lower_layer_enc_height 0.1 \  
-upper_layer_enc_width 0.01 \  
-upper_layer_enc_height 0.01 \  
-min_cut_spacing 0.25
```

Using parameters always creates a symmetric, single-cut via master.

The following command creates an H-shaped via master by specifying a geometry list:

```
prompt> create_via_master \  
-name VIA1 \  
-cut_layer_name CUT \  
-lower_layer_name METAL1 \  
-upper_layer_name METAL2 \  
-rectangles { \  
  0.22 0.22 0.1 0.1 0.01 0.01 0.25
```

```
{METAL1 {-0.130 -0.035 0.130 0.035}} \
{CUT {0.035 -0.035 0.100 0.035}} \
{CUT {-0.100 -0.035 -0.035 0.035}} \
{METAL2 {-0.100 -0.065 -0.035 0.065}} \
{METAL2 {0.035 -0.065 0.100 0.065}} \
}
```

SEE ALSO

`create_via(2)`

create_via_rule

Creates a via_rule.

SYNTAX

```
collection create_via_rule  
  -name via_rule_name  
  [-cut_layer_names {name_list}]  
  [-cut_names {name_list}]  
  [-cut_rows {int_list}]  
  [-cuts_per_row {int_list}]
```

Data Types

<i>name_list</i>	cut_names
<i>int_list</i>	cuts_per_row

ARGUMENTS

-name *via_rule_name*

Specifies the name of the via_rule to be created.

-cut_layer_names {*name_list*}

Defines the cut layer's names. This is must for defining a via_ladder. If specified then other via_ladder related attributes are must and length of this list attribute should be same as of other via_ladder related attributes.

-cut_names {*name_list*}

Defines the cut table names. This is must for defining a via_ladder. If specified then other via_ladder related attributes are must and length of this list attribute should be same as of other via_ladder related attributes.

-cut_rows {*int_list*}

Defines the number of cut rows. This is must for defining a via_ladder. If specified then other via_ladder related attributes are must and length of this list attribute should be same as of other via_ladder related attributes.

-cuts_per_row {*int_list*}

Defines the number of cuts per row. This is must for defining a via_ladder. If specified then other via_ladder related attributes are must and length of this list attribute should be same for other via_ladder related attributes.

RETURN VALUE

The command returns the created via_rule (as a collection), an empty string if it fails, or a TCL_ERROR if there is a command syntax error.

DESCRIPTION

This command creates a new via_rule.

EXAMPLES

The following example creates a via_rule in current lib.

```
prompt> create_via_rule -name VR1
```

The following example creates a via_ladder type of via_rule.

```
prompt> create_via_rule -name VL1 -cut_layer_names {VIA1 VIA2} \ cut_names {cut1 cut2} -cut_rows {2 2} -cuts_per_row {2 2}
```

SEE ALSO

get_via_rules(2)

create_voltage_area

Creates a voltage area at a specified region that constrains the placement of specified hierarchical cells. This command is supported only in topographical mode.

SYNTAX

```
int create_voltage_area
    modules -name voltage_area_name
    | -power_domain {power_domain_name}
    -coordinate {llx1 lly1 urx1 ury1 ...}
    [-guard_band_x guard_band_width]
    [-guard_band_y guard_band_width]
    [-is_fixed]
    [-color color | -cycle_color]
    [-target_utilization float]
```

Data Types

```
modules          list
voltage_area_name string
power_domain_name string
llx1             float
lly1             float
urx1             float
ury1             float
guard_band_width float
color            string
```

ARGUMENTS

modules

Specifies a list of hierarchical cells that are constrained to be placed in the specified area. A given cell cannot be assigned to two different voltage areas. Leaf-level cells (standard cells) cannot be specified in this option.

In UPF mode, voltage areas may only be created if they contain all of the hierarchical cells from one or more power domains. The **-power_domain** argument should be used instead of this argument.

-name voltage_area_name

Specifies the name of the voltage area to be created. It cannot be the name of an existing voltage area or the name "DEFAULT_VA", which is reserved for the default voltage area.

If this option is used with the **-power_domain** option, you can put multiple compatible power domains into the same voltage area. Otherwise, this option can be used only with the *modules* argument.

-power_domain {*power_domain_name*}

Creates a voltage area from an existing power domain previously created by the **create_power_domain** command. By default, the name of the new voltage area is identical to the name of the power domain. All hierarchical cells in the power domain are associated with the voltage area. After the voltage area is successfully created, the power domain contains the corresponding boundary information. You cannot create a voltage area from a top-level power domain.

-coordinate {*llx1 lly1 urx1 ury1 ...*}

Specifies a list of lower-left and upper-right coordinates of a voltage area, in microns. Each set of {*llx lly urx ury*} defines a target rectangular placement area. The voltage area geometry can be a rectangle or a rectilinear polygon.

-guard_band_x *guard_band_width*

Specifies the horizontal (left and right) margin around the voltage area where objects are not allowed to be placed, a positive floating-point value.

-guard_band_y *guard_band_width*

Specifies the vertical (top and bottom) margin around the voltage area where objects are not allowed to be placed, a positive floating-point value.

-is_fixed

Specifies that the voltage area is in a fixed location and cannot be changed by shaping. By default, the voltage area can be changed by shaping.

-color *color*

Specifies the color for the voltage area and its cells in the GUI display. The value can be either a color string such as "red" or "blue" or a color index, an integer between 0 to 63, each number representing a different color.

-cycle_color

Automatically chooses the next color in the color table for successive new voltage areas. This option is mutually exclusive with the **-color** option.

-target_utilization *float*

Specifies the target utilization for this voltage area during shaping. The target voltage area is the total area of cells placed in the voltage area divided by the target utilization for the voltage area. The default is the design utilization. The value specified must be between 0.1 and 1.0.

DESCRIPTION

The **create_voltage_area** command creates a voltage area of a specific geometry and location on the core area of the chip. The voltage area is associated with a list of one or more power domains previously created by the **create_power_domain** command or a list of specified hierarchical cells. The placer considers the voltage area an exclusive, hard move bound; it tries to place all the cells associated with the voltage area within the that area, and to place all other cells outside of that area.

Voltage areas can be physically nested, with one voltage area completely enclosing another voltage area. When considering the area or utilization of the outside voltage area, the area of inner voltage area is excluded.

Voltage areas can also be logically nested, with one hierarchical cell belonging to one voltage area while one or more of its descendants belonging to other voltage areas. When considering area utilization, the tool excludes all descendants belonging to other voltage areas.

Any nested voltage area must be fully enclosed by another voltage area; partially overlapping voltage areas are not allowed. Any move bound must be completely enclosed inside a voltage area; a voltage area cannot partially overlap or be enclosed by a move bound.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example constrains the hierarchical instance INST_1 to lie within the voltage area whose coordinates are lower-left corner (100,100) and upper-right corner (200,200):

```
prompt> create_voltage_area -name VA1 \  
-coordinate {100 100 200 200} INST_1
```

The following example creates a voltage area of disjoint polygons using the **-coordinate** option:

```
prompt> create_voltage_area -power_domain PD1 -coordinate \  
{ { {100 100} {300 100} {300 200} {200 200} } \  
{200 300} {100 300} {100 100} } \  
{ {400 400} {500 400} {500 500} {400 500} {400 400} } }
```

Each power domain must be fully contained within a single voltage area. To restrict placement of elements that are a subset of a power domain, see the **create_bounds** command.

SEE ALSO

create_power_domain(2)
remove_voltage_area(2)
report_voltage_area(2)
create_bounds(2)

create_wiring_keepouts

Creates a wiring keepout.

SYNTAX

```
status create_wiring_keepouts  
-name name  
-layer layer  
-coordinate list_of_float_values
```

Data Types

<i>name</i>	string
<i>layer</i>	string
<i>list_of_float_values</i>	list

ARGUMENTS

-name *name*

Specifies the name of the wiring keepout to be created.

-layer *layer*

Specifies the name of the layer on which the routing obstruction is to lie.

-coordinate *list_of_float_values*

Specifies a list of float values that serve as the coordinates of the wiring keepout. These coordinates are relative to the design origin at (0 0) and are in units of micron.

DESCRIPTION

The **create_wiring_keepouts** command creates a wiring keepout.

A wiring keepout is not created if the given keepout name matches with any other existing wiring keepout name. However, a wiring keepout is created if its name matches the name of an existing placement keepout.

EXAMPLES

The following is an example of the **create_wiring_keepouts** command:

```
prompt> create_wiring_keepouts -name "my_keep1" -layer "METAL1" \  
      -coord {12 12 100 100}
```

SEE ALSO

extract_physical_constraints(2)
report_physical_constraints(2)
reset_physical_constraints(2)

cross_probing_filter

Filters a collection based on file name and line number criteria.

SYNTAX

```
collection cross_probing_filter  
-source source_patterns  
objects
```

Data Types

<i>source_patterns</i>	string
<i>objects</i>	collection

ARGUMENTS

-source *source_patterns*

This string is the matching criteria used for filtering. Filter a collection by specifying a source file and line number in the "file_pattern[:line_pattern]" format. You can specify multiple patterns by delimiting them with a space character, such as "file_pattern1[:line_pattern1] file_pattern2[:line_pattern2] ...". Objects that match any of the specified file/line patterns are returned in the resulting collection.

The asterisk (*) and question mark (?) wildcard patterns are allowed in the pattern.

objects

Specifies the base collection to be filtered. The base collection may contain cell or port objects.

DESCRIPTION

Allows you to filter a collection by specifying a source file and line number. This command functions similar to **filter_collection** except the filtering criteria is specific to cross-probing.

You must have a design that was analyzed and elaborated or synthesized using Design Compiler version I-2013.12 or later. The tool provides information about where the cell or port was created, specifying which file and line number the object came from. These objects can be filtered based on file name and/or line number.

Some objects in the design might have been merged, resulting in an object that can potentially have multiple source locations. In this case, the object is returned in the result collection if any of its source positions match any of the specified source patterns specified by *source_patterns*.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the result from the **cross_probing_filter** command, returning cells that originated from a source file ending in m.v, and on line 21:

```
prompt> cross_probing_filter [get_cells -hierarchical] -source "**m.v:21"  
{mid1/bot1/c1 mid1/bot2/c1}
```

SEE ALSO

report_cross_probing(2)
get_cross_probing_info(2)

current_design

Sets the working design.

SYNTAX

string **current_design**
[*design*]

Data Types

design string

ARGUMENTS

design

Specifies the working or focal design for many commands.

If you do not specify a design or specify a period (.), the tool returns the current working design. If you specify a design that cannot be found, the tool issues an error and the working design remains unchanged.

DESCRIPTION

The **current_design** command sets the working design for many commands. If you do not specify any arguments, the **current_design** command returns the name of the current working design.

To display designs currently available in memory, use the **list_designs** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **current_design** command to show the current context and to change the context from one design to another:

```
prompt> current_design .
```

```
Current design is 'TOP'.  
{TOP}
```

```
prompt> list_designs  
ADDER      FULL_SUBTRACTOR HALF_SUBTRACTOR TOP (*)  
FULL_ADDER  HALF_ADDER    SUBTRACTOR
```

```
prompt> current_design ADDER  
Current design is 'ADDER'.  
{ADDER}
```

```
prompt> current_design  
Current design is 'ADDER'.  
{ADDER}
```

You can use the **current_design** command as an argument to other commands. In the following example, the **remove_design** command is used to delete the working design from memory:

```
prompt> current_design  
Current design is 'TOP'.
```

```
prompt> remove_design [current_design]  
Removing design 'TOP'
```

```
prompt> current_design  
Error: Current design is not defined. (UID-4)
```

SEE ALSO

```
current_instance(2)  
list_designs(2)  
list_instances(2)
```

current_design_name

Returns the current design name.

SYNTAX

string **current_design_name**

DESCRIPTION

This command returns the string of the current design name.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **current_design_name** to save the current design name to a variable:

```
prompt> current_design  
Current design is 'TOP'.  
{TOP}
```

```
prompt> set design_name [current_design_name]  
Information: Defining new variable 'design_name'. (CMD-041)  
TOP  
Current design is 'TOP'.
```

SEE ALSO

current_design(2)

current_dft_partition

Sets or gets the design partition for the current specification.

SYNTAX

status **current_dft_partition**
[*design_partition_label*]

Data Types

design_partition_label string

ARGUMENTS

design_partition_label

Specifies the working design partition for many commands. If the *design_partition_label* is not specified, the tool uses the last design partition specified. If the *design_partition_label* specifies a partition name that is not defined in the CTL model for the current design and that has not been specified previously using the **define_dft_partition** command, an error is issued and the working test mode remains unchanged.

DESCRIPTION

The **current_dft_partition** command sets the design partition for many specification commands, such as **set_dft_signal**, **set_scan_configuration**, **set_scan_compression_configuration**, and **set_scan_path**. When no arguments are specified, **current_dft_partition** returns the name of the current DFT partition.

To display the names of all of the design partitions in the test model for the current design, use the **report_dft_partition** command.

EXAMPLES

The following example sets the required number of scan chains to 3 and 2 to the partitions *part1* and *part2*, respectively:

```
prompt> define_dft_partition part1 -include {U1 U2}
```

```
prompt> define_dft_partition part2 -include {U3 U4}
```



```
prompt> current_dft_partition part1

prompt> set_scan_configuration -chain_count 3

prompt> current_dft_partition part2

prompt> set_scan_configuration -chain_count 2

prompt> preview_dft
1
```

SEE ALSO

- `define_dft_partition(2)`
- `insert_dft(2)`
- `preview_dft(2)`
- `remove_dft_partition(2)`
- `report_dft_partition(2)`
- `set_dft_signal(2)`
- `set_scan_compression_configuration(2)`
- `set_scan_configuration(2)`
- `set_scan_path(2)`

current_instance

Sets the working instance object and enables other commands to be used on a specific cell in the design hierarchy.

SYNTAX

```
string current_instance  
    [instance]
```

Data Types

instance string

ARGUMENTS

instance

Specifies the working cell.

The **current_instance** command operates with a variety of *instance* arguments:

- If you do not specify the *instance* argument, the focus is returned to the top level of the hierarchy.
- If you specify ".", the current instance is returned and no change is made.
- If you specify "..", the current instance is moved up one level in the design hierarchy. You can also nest the ".." directive in complex instance specifications. For example, ../../MY_INST attempts to move the context up two levels of hierarchy, and then down one level to the MY_INST cell.
- If you specify a valid cell at the current level of the hierarchy, the current instance is moved down to that level of the design hierarchy.
- You can traverse multiple levels of hierarchy in a single call to the **current_instance** command by separating multiple cell names with slashes. For example, if you specify U1/U2, the current instance is moved down two levels of hierarchy if both cells exist at the current levels in the design hierarchy.

More complex examples of *instance* arguments are described below in EXAMPLES.

DESCRIPTION

The **current_instance** command sets the working instance. An instance is a cell embedded in the hierarchy of a design. Usually you define an instance to set or get attributes on a cell.

To display the instances available at the current level of a design hierarchy, use the **list_instances** command.

The **current_design** command changes the working design, setting the current instance to the top level of the new current design.

The **current_instance** command traverses the design hierarchy similar to the way the UNIX **cd** command traverses the file hierarchy.

Note that the **current_instance** command does not work on leaf cells. If you attempt to run the **current_instance** command on a leaf cell, an error occurs.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **current_instance** command to move up and down the design hierarchy and the **list_instances** command to show the available instances at each point in the hierarchy:

```
prompt> current_design .  
Current design is 'TOP'.  
{TOP}
```

```
prompt> list_instances  
U1 (ADDER)    U2 (SUBTRACTOR)
```

```
prompt> current_instance U1  
Current instance is 'TOP/U1'.  
/TOP/U1
```

```
prompt> current_instance .  
Current instance is 'TOP/U1'.  
/TOP/U1
```

```
prompt> list_instances  
U1 (FULL_ADDER) U2 (FULL_ADDER) U3 (FULL_ADDER) U4 (FULL_ADDER)
```

```
prompt> current_instance U3  
Current instance is 'TOP/U1/U3'.  
/TOP/U1/U3
```

```
prompt> current_instance "../U4"  
Current instance is 'TOP/U1/U4'.  
/TOP/U1/U4
```

```
prompt> current_instance  
Current instance is the top-level of design 'TOP'.
```

In the following example, changing the current design resets the **current_instance** to the top level of the new design hierarchy:

```
prompt> current_design  
Current design is 'TOP'.  
{TOP}
```

```
prompt> current_instance "U2/U1"  
Current instance is 'TOP/U2/U1'.  
/TOP/U2/U1
```

```
prompt> current_design ADDER  
Current design is 'ADDER'.  
{ADDER}
```

```
prompt> current_instance .  
Current instance is the top-level of design 'ADDER'.
```

The following example uses **current_instance** to go to an instance of another design. The current design is set by the new design whose name is the name after the first slash of the given instance name.

```
prompt> current_design .  
Current design is 'TOP'.  
{TOP}
```

```
prompt> current_instance U1  
Current instance is 'TOP/U1'.  
/TOP/U1
```

```
prompt> current_instance /TOP/U2  
Current instance is 'TOP/U2'.  
/TOP/U2
```

```
prompt> current_instance /ALARM_BLOCK/U6  
Current instance is 'ALARM_BLOCK/U6'.  
/ALARM_BLOCK/U6
```

SEE ALSO

- `current_design(2)`
- `list_designs(2)`
- `list_instances(2)`

current_mw_lib

Gets the current Milkyway library.

SYNTAX

collection **current_mw_lib**

ARGUMENTS

The **current_mw_lib** command has no arguments.

DESCRIPTION

The **current_mw_lib** command gets the current Milkyway library. The **current_mw_lib** command returns a collection of the current Milkyway library, if one exists.

Many Milkyway library-related commands use the current Milkyway library by default. The current Milkyway library is automatically set when the **open_mw_lib** command opens a Milkyway library to use.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns the current Milkyway library in the current session:

```
prompt> current_mw_lib  
{design}
```

SEE ALSO

close_mw_lib(2)

copy_mw_lib(2)
create_mw_lib(2)
open_mw_lib(2)
rebuild_mw_lib(2)
rename_mw_lib(2)
report_mw_lib(2)

current_scenario

Sets the current scenario.

SYNTAX

```
string current_scenario  
[scenario_name]
```

Data Types

```
scenario_name  string
```

ARGUMENTS

scenario_name

Specifies the name of the current scenario. If *scenario_name* is not specified, the tool returns to the current scenario.

DESCRIPTION

The **current_scenario** command sets the current scenario for many commands. Without arguments, **current_scenario** returns the name of the current scenario.

The creation and usage of scenarios is available with Design Compiler Graphical.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following example uses the **current_scenario** command to set the current scenario:

```
prompt> create_scenario MODE1  
Warning: Discarding all scenario-specific information previously defined  
in this session. (UID-1008)  
Current scenario is: MODE1  
1
```

```
prompt> create_scenario MODE2  
Current scenario is: MODE2  
1
```

```
prompt> create_scenario MODE3  
Current scenario is: MODE3  
1
```

```
prompt> current_scenario MODE2  
Current scenario is: MODE2  
MODE2
```

SEE ALSO

```
all_scenarios(2)  
create_scenario(2)  
remove_scenario(2)
```

current_test_mode

Sets or gets the current working test mode for the current design.

SYNTAX

status **current_test_mode**
[*test_mode_label*]

Data Types

test_mode_label string

ARGUMENTS

test_mode_label

Specifies the working test mode for many commands.

If the *test_mode_label* is not specified, the tool prints the current test mode.

DESCRIPTION

The **current_test_mode** command sets the current working mode. This is the default test mode used for many DFT commands, such as **set_dft_signal** and **dft_drc**.

To display the names of all the modes in the test model data for the current design, use the **report_test_model** command.

To display the names of all the test modes of the current design defined with the **define_test_mode** command, use the **list_test_modes** command.

EXAMPLES

The following example sets the test mode to "Internal_scan" for the current design:

```
prompt> current_test_mode Internal_scan  
1
```

The following example gets the test mode for the current design:

```
prompt> current_test_mode  
Current test mode is 'Internal_scan'.  
1
```

SEE ALSO

current_design(2)
dft_drc(2)
list_test_modes(2)
report_test_model(2)

cut_row

Removes rows from the current design.

SYNTAX

status **cut_row**
[-all | -within {*coordinates*}

Data Types

coordinates list

ARGUMENTS

-all

Removes all rows from the current design. This option and the **-within** option are mutually exclusive.

-within {*coordinates*}

Removes all rows in the specified region. For a rectangular region, specify the coordinates for the lower-left and upper-right corners as {{*ll_x ll_y*} {*ur_x ur_y*}}. For a rectilinear region, specify a list of rectilinear coordinates in either clockwise or counterclockwise order. This option and the **-all** option are mutually exclusive.

This option is not supported in Design Compiler topographical mode and will be ignored.

DESCRIPTION

This command removes rows from the current design. If you specify the **-all** option, the command removes all rows from the design. If you specify the **-within** option, the command removes rows within the specified region.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **-all** option to remove all rows from the current design.

```
prompt> cut_row -all  
1
```

SEE ALSO

add_row(2)

date

Returns a string containing the current date and time.

SYNTAX

string **date**

DESCRIPTION

The **date** command generates a string containing the current date and time, and returns that string as the result of the command. The format is fixed as follows:

ddd mmm nn hh:mm:ss yyyy

Where:

ddd is an abbreviation for the day
mmm is an abbreviation for the month
nn is the day number
hh is the hour number (24 hour system)
mm is the minute number
ss is the second number
yyyy is the year

The **date** command is useful because it is native. It does not fork a process. On some operating systems, when the process becomes large, no further processes can be forked from it. With it, there is no need to call the operating system with **exec** to ask for the date and time.

EXAMPLES

The following command prints the date.

```
prompt> echo "Date and time: [date]"  
Date and time: Thu Dec 9 17:29:51 1999
```

SEE ALSO

exec(2)

dc_allocate_budgets

Allocates budgets to specified cells.

SYNTAX

```
string dc_allocate_budgets
  [cell_list]
  [-write_script]
  [-file_format_spec format_spec]
  [-format dctcl | dcsh]
  [-separator hierarchy_separator]
  [-mode rtl | gate | mixed]
  [-budget_design_ware]
  [-levels budget_levels]
  [-no_interblock_logic]
  [-min_register_to_output minimum_budget]
  [-min_input_to_output minimum_budget]
  [-min_input_to_register minimum_budget]
```

Data Types

<i>cell_list</i>	list
<i>format_spec</i>	string
<i>hierarchy_separator</i>	string
<i>budget_levels</i>	string
<i>minimum_budget</i>	float

ARGUMENTS

cell_list

Specifies the list of cells to budget. The list of cells is required when you do not use the **-levels** option.

-write_script

Specifies that the context of each budgeted cell or design is to be written into a constraint file.

-file_format_spec *format_spec*

Specifies the format for the names of constraint files generated for each budgeted cell, if the **-write_script** option is used. The default is */%C.dctcl*. For details, see the DESCRIPTION section.

-format *dctcl* | *dcsh*

Specifies the output format for script files. Allowed values are **dctcl** (the default) or **dcsh**.

-separator *hierarchy_separator*

Specifies a single character to use as a separator in deriving the names of constraint files generated for each cell. All characters are allowed except slash (/), percent (%), and double backslash \\. The default is the at sign (@).

-mode *rtl | gate | mixed*

Specifies whether the design to be budgeted is an RTL or a gate-level design, or a design that is mixed with RTL and gates. The default is **gate**. For details on the **-mode** option, see the *Budgeting for Synthesis User Guide*.

-budget_design_ware

Specifies that DesignWare components are to be budgeted. By default, Design Budgeter considers these components fixed.

-levels *budget_levels*

Specifies the number of levels of hierarchy for which to allocate budgets starting from the specified cells. Allowed values are **all**, **1**, **2**, **3**..., and **n**. The default is **all**, which allocates budgets down all levels of hierarchy.

-no_interblock_logic

Specifies that interblock logic is not to be budgeted. By default, the design budgeter considers interblock logic not fixed.

-min_register_to_output *minimum_budget*

Specifies a positive floating-point number that indicates the minimum budget for any path segment from an internal register to an output pin of a budgeted cell.

-min_input_to_output *minimum_budget*

Specifies a positive floating-point number that indicates the minimum budget for any path segment from an input pin to an output pin of a budgeted cell.

-min_input_to_register *minimum_budget*

Specifies a positive floating-point number that specifies the minimum budget for any path segment from an input pin to an internal register of a budgeted cell.

The **-min_register_to_output**, **-min_input_to_output**, and **-min_input_to_register** options allow you to avoid overconstraining noncompressible path segments, such as segments to registered output pins or segments from registered input pins. Use these options carefully to avoid creating mutually inconsistent constraints. If necessary, the budgeter assigns path delays less than these minimum budget values to prevent negative slack in the budget.

DESCRIPTION

The **dc_allocate_budgets** command distributes the timing constraints among the specified cells. The command allows you to specify cells by either non-hierarchical budgets allocation or the hierarchical budgets allocation.

The non-hierarchical budgets allocation is the default method. Only cells specified with *cell_list* are budgeted. Designs are not budgeted.

The hierarchical budgets allocation is enabled by the **-levels** option. When activated, the tool allocates budgets hierarchically. If you specify *cell_list*, budgets are hierarchically allocated starting from these cells. If you do not specify *cell_list*, budgets are allocated starting from cells in the top level of the design. If a design has several instances, a constraint file can be written out for each instance for cell-based file format specification (%C). For design-based file format specification (%D), the last instance of the design is used to generate the constraint file.

The **-file_format_spec** option specifies the format for the names of constraint files generated for each budgeted cell. The format contains the directory in which files are to be written, and the prefix, the suffix, and the extension to use. The format specification must contain only one conversion specification: either %D, which indicates a design name, or %C, which indicates a cell name.

For the %D conversion specification, the constraint file name is obtained by replacing the conversion specification by either the name of the cell design if the design is referenced once (uniquified design), or by the concatenation of the design name and the full cell name.

For the %C conversion specification, the constraint file name is obtained by replacing the conversion specification by the full cell name.

The default value of **-file_format_spec** depends on the **-format** option.

The default value is `./%C.dctcl` if the **-format** option is not specified, `./%C.dctcl` if the **-format** option value is **dctcl**, or `./%C.dcsch` if the **-format** option is **dcsch**.

By default, budget allocation is performed automatically. You can optionally specify user budgets or budget ratios using the **set_user_budget** command.

EXAMPLES

The following command allocates budgets to cells *I1* and *I2*, and writes a constraint file in dcsch format to files named *I1.dcsch* and *I2.dcsch*. The report shows the results.

```
prompt> dc_allocate_budgets -write_script \
      -file_format_spec %C.dcsch -format dcsch {I1 I2}
```

```
prompt> report_path_budget
```

```
Loading design 'budtest1'
```

```
*****
Report : budget
  -path full
  -delay max
  -max_paths 1
Design : budtest1
Version: 2000.11-PROD
Date   : Mon Oct 16 11:12:16 2000
*****
```

```
Operating Conditions:
Wire Load Model Mode: top
```

```
Startpoint: I1/reg3 (rising edge-triggered flip-flop clocked by clk)
Endpoint: I3/reg1 (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: max
```

Point	Incr	Path
clock clk (rise edge)	0.00	0.00
clock network delay (ideal)	0.00	0.00
I1/reg3/CP (FD1)	0.00	0.00 r

I1/reg3/Q (FD1)	0.85 *	0.85 f	
I1/out2 (desA)	0.30	1.15 f	
I2/in2 (desB)	0.00	1.15 f	
I2/out2 (desB)	1.75	2.90 f	
I3/in2 (desC)	0.00	2.90 f	
I3/reg1/D (FD1)	0.30	3.20 f	
data arrival time		3.20	
clock clk (rise edge)	4.00	4.00	
clock network delay (ideal)	0.00	4.00	
I3/reg1/CP (FD1)	0.00	4.00 r	
library setup time	-0.80	3.20	
data required time		3.20	

data required time		3.20	
data arrival time		-3.20	

slack (VIOLATED: increase significant digits)			0.00

1

SEE ALSO

report_path_budget(2)
set_user_budget(2)

decrypt_lib

Decrypts a Milkyway library containing locked technology information.

SYNTAX

```
status decrypt_lib
[-format {mwlib}]
-key key_string
lib_name
```

Data Types

<i>key_string</i>	string
<i>lib_name</i>	string

ARGUMENTS

-format {*mwlib*}

Specifies the format of the file to be decrypted. By default, the format is **mwlib**, which means Milkyway library.

-key *key_string*

Specifies the key string to unlock the Milkyway library technology information. The library can be decrypted only by supplying the correct key string.

lib_name

Specifies the name of the Milkyway library to decrypt.

DESCRIPTION

This command can decrypts an encrypted mw library if the library was encrypted by command "**encrypt_lib -format mwlib -key *keyStr***".

A library encrypted natively by creation from an encrypted technology file cannot be decrypted by this command.

EXAMPLES

The following example decrypts the Milkyway library "my_lib" with a key string.

```
prompt> decrypt_lib -key {ki08$#98fa?ipo} my_lib
```

SEE ALSO

`encrypt_lib(2)`

define_design_lib

Maps a design library to a UNIX directory.

SYNTAX

```
status define_design_lib  
      library_name  
      -path directory
```

Data Types

```
library_name  string  
directory     string
```

ARGUMENTS

library_name

Specifies the library to be mapped.

-path *directory*

Specifies the directory to which the library is mapped.

DESCRIPTION

The **define_design_lib** command maps a design library to a UNIX directory. The directory is used to store intermediate representations of designs.

When **define_design_lib** is used with the **design_library_file** command, the file is automatically reread when it is changed. As a result, if the file contains a library definition that was manually defined with **define_design_lib**, the manual definition is overridden.

EXAMPLES

The following example uses **define_design_lib** to map the *MY_LIB* library to the *~library* directory:

```
prompt> define_design_lib MY_LIB -path ~/library
```

SEE ALSO

analyze(2)
elaborate(2)
get_design_lib_path(2)
read(2)
report_design_lib(2)
write_design_lib_paths(2)

define_dft_design

Characterizes a design as a DFT design to be used for DFT insertion.

SYNTAX

```
status define_dft_design
  -design_name design_name
  [-type design_type]
  [-interface access_list]
  [-test_model model_path_name]
  [-params param_list]
  [-validate true | false]
```

Data Types

```
design_name      string
design_type      string
access_list     list
model_path_name string
param_list      list
```

ARGUMENTS

-design_name *design_name*

Identifies the design to be instantiated during DFT insertion.

This option is required.

-type *design_type*

Specifies the type of the DFT design to which the design corresponds. Valid design types are:

```
DECODE      CONTROL_FORCE  Z_CONTROL_FORCE  OBSERV_DGEN
SHIFT_REG   BC_1           BC_2           BC_3
BC_4        BC_5           BC_7           BC_8
BC_9        BC_10         WC_D1           WC_D1_S
WC_S1       WC_S1_S       INSTRREG      TAP
TAP_UC      LBIST_CONTROLLER LBIST_CODEEC  LBIST_XCONTROLLER
LBIST_XCODEC CLK_MUX      CLK_CHAIN    MBIST_CONTROLLER
MBIST_WRAPPER PAD         AC_1         AC_2
AC_7        AC_SELU      AC_SELX
```

There is no default for this option. When the option is not specified, the tool creates a new DFT design type from the specified design name and the specified interface access list.

-interface *access_list*

Specifies the list of signal mapping triplets relating a signal type to a pin of the design specified with **-design_name** *design_name*. The valid format of a signal mapping triplet is as follows:

signal_type_name pin_name pin_polarity

The *signal_type_name* specifies the signal type name of the latest of DesignWare version of the DFT design pin name. The *pin_name* specifies the pin name of the design specified with **-design_name** *design_name*. The *pin_polarity* specifies the polarity of the specified design pin with respect to the *signal_type_name*.

Valid values for *pin_polarity* are **h** for normal polarity and **l** for negative polarity.

There is no default for this option. When the option is not specified, it is assumed that the design does not have interface access pins for use with DFT insertion.

-test_model *model_path_name*

Identifies the path name of the test model for the specified DFT design.

There is no default for this option.

-params *param_list*

Specifies the list of parameter triplets for the specified DFT design. The valid format of a parameter triplet is as follows:

param_name param_type param_value

The *param_name* specifies the name of the parameter, *param_type* specifies the type of the parameter, and *param_value* specifies the value of the parameter. Valid values for *param_type* are **integer**, **float**, **boolean**, and **string**.

These parameters are stored on the specified DFT design for use with DFT insertion.

There is no default for this option.

-validate **true | **false****

Validates connectivity in netlist DFT designs.

When set to **true**, the **define_dft_design** command validates that each output/inout port in the interface specification reaches at least one input port in the interface specification. Port direction is determined by the design netlist specified with the **-design** option. The check is a simple topological fanin check that does not analyze cell functionality or design constants. If the validation fails, the tool issues an error and discards the specification.

The default is **false** so that specifications relying on inferred ports do not cause validation failures.

This option is considered only for design netlist representations; it is ignored for black box or library cell representations.

DESCRIPTION

The **define_dft_design** command characterizes a design as a DFT design of a certain type for use with DFT insertion. Ensure that the specified design functionality matches the standard design of the specified type. The specified design is used to instantiate an IP instead of a DesignWare IP during DFT insertion.

EXAMPLES

The following example instructs **insert_dft** to use the *my_des* design as a WC_D1 for wrapper insertion:

```
prompt> read_db my_des.db
1

prompt> define_dft_design -design_name my_des -type WC_D1 \
    -interface {shift_clk capture_clk h capture_en capture_en \
        h shift_en shift_dr h cti si h cto so h cfi data_in \
        h cfo data_out h}
1
```

The following example instructs **insert_dft** to use the *my_des1* design as a BC_4 for BSD insertion:

```
prompt> read_db my_des1.db
1

prompt> define_dft_design -design_name my_des1 -type BC_4 \
    -interface {capture_clk cap_clk h capture_en cen h \
        shift_dr shift h si ti h so to h data_in di h data_out do h}
1
```

The following example instructs **insert_dft** to use the *ip_pad_des1* design as an input pad design for BSD insertion:

```
prompt> read_db ip_pad_des1.db
1

prompt> define_dft_design -design_name ip_pad_des1 -type PAD \
    -interface {port di h data_out do h} \
    -params {$pad_type$ string input}
1
```

The following example instructs **insert_dft** to use the *op_pad_des1* design as a tristate output pad design with an inverted enable pin for BSD insertion:

```
prompt> read_db op_pad_des1.db
1

prompt> define_dft_design -design_name op_pad_des1 -type PAD \
    -interface {port do h data_in di h enable en l} \
    -params {$pad_type$ string tristate_output}
1
```

The following example instructs **insert_dft** to use the *bidi_pad_des1* design as a bidirectional differential pad design library cell with observe/high/low pins for BSD insertion:

```
prompt> read_db bidi_pad_des1.db
1

prompt> define_dft_design -design_name bidi_pad_des1 -type PAD \
    -interface {port ZP h port ZM l data_in di h \
        data_out do h observe IE h high TM h low TE h} \
    -params {$pad_type$ string bidirectional \
        $differential$ string true $lib_cell$ string true}
1
```

1

The following example instructs **insert_dft** to use the *ip_op_pad_des1* design as an differential input as well as a differential output pad design for BSD insertion:

```
prompt> read_db ip_op_pad_des1.db
1

prompt> define_dft_design -design_name ip_op_pad_des1 -type PAD \
    -interface {port dop h port don l data_in di h} \
    -params {$pad_type$ string output $differential$ string true}
1

prompt> define_dft_design -design_name ip_op_pad_des1 -type PAD \
    -interface {port dip h port din l data_out do h} \
    -params {$pad_type$ string input $differential$ string true}
1
```

The following example instructs **insert_dft** to use the *ip_op_pad_des1* design as a differential hybrid pad design for BSD insertion:

```
prompt> read_db ip_op_pad_des1.db
1

prompt> define_dft_design -design_name ip_op_pad_des1 -type PAD \
    -interface {si si_pin H so so_pin H shift_dr shift_dr_pin H} \
    -params {$pad_type$ string hybrid $differential$ string true \
    $diff_port_pairs$ list string \
    "dop don" "dip din" $end_list$ \
    $bsr_segment$ list string \
    "0 BC_1 - dop X -" \
    "1 BC_4 dip - X -" "2 BC_2 dip - X -" "3 BC_4 din - X -" \
    $end_list$}
1
```

SEE ALSO

```
insert_dft(2)
preview_dft(2)
remove_dft_design(2)
report_dft_design(2)
```

define_dft_partition

Defines a design DFT partition to be created during DFT synthesis.

SYNTAX

```
status define_dft_partition
    design_partition_label
    [-default false | true]
    [-include list_of_design_objects]
    [-clocks list_of_clocks]
    [-rising_edge_clocks list_of_clocks]
    [-falling_edge_clocks list_of_clocks]
    [-extest_cells list_of_wrapped_cores]
    [-wrapper enable | disable]
```

Data Types

```
design_partition_label    string
list_of_design_objects   list
list_of_clocks          list
list_of_wrapped_cores   list
```

ARGUMENTS

design_partition_label

Specifies a user-defined text string containing only alphanumeric characters (a-z, A-Z, 0-9) and the underscore (_).

-default false | true

Specifies whether the defined partition is also the default partition.

By default, the tool creates a default partition named **default_partition** for any design logic not explicitly included in a user-defined partition.

If you set this option to **true**, this defined partition is the default partition. Any unallocated logic is included in this partition along with any design logic explicitly included using other options of this command.

Only one partition can be defined as the default partition.

-include *list_of_design_objects*

Specifies the list of leaf or hierarchical cells, design references, or core scan segments that belong to the defined partition.

In DFTMAX shared codec I/O flows, you can also specify subpartition names to define sharing groups. Partition names are not supported in other flows.

You can mix object types in the list. Wildcards and collections are supported.

-clocks *list_of_clocks*

Specifies the list of clocks that belongs to the defined partition.

-rising_edge_clocks *list_of_clocks*

Specifies the list of rising clocks that belongs to the defined partition.

-falling_edge_clocks *list_of_clocks*

Specifies the list of falling clocks that belongs to the defined partition.

-extest_cells *list_of_wrapped_cores*

Specifies that the outward-facing wrapper chains from the specified cores should be included in the defined partition.

The referenced cores can exist in other DFT partitions. This option provides more control over how the outward-facing chains of wrapped cores are compressed by a top-level codec in a DFT partition flow.

This option affects all test modes that use the specified cores in outward-facing modes, including uncompressed scan modes. Test modes that use the cores in inward-facing modes are unaffected by this option.

DESCRIPTION

The **define_dft_partition** command allows you to define a specific design partition during DFT synthesis. The partition is referenced to the top level in a top-down flow.

The partition information is then used during the DFT architect and insertion process, while running the **preview_dft** or the **insert_dft** command. Each design partition will be DFT-inserted based on its DFT constraints.

You can use multiple option types (**-include**, **-clocks**, **-rising_edge_clocks**, and **-falling_edge_clocks**) in a single partition definition.

You can include clock-gating cells in DFT partition definitions. Although clock-gating cells are not stitched into scan chains, they will use partition-specific scan-enable signals for their test pins (if defined). Clock-gating cells can be included by leaf cell, enclosing hierarchical cell, or clock. When referencing a Power Compiler clock-gating cell by leaf cell, reference the mapped clock-gating cell inside the hierarchical clock-gating cell wrapper.

EXAMPLES

The following example defines 2 design partitions for use in a DFT insertion. The design contains at the top level 4 hierarchical cells named *U1*, *U2*, *U3* and *U4*, respectively.

```
prompt> define_dft_partition part_1\  
-include {U1 U2}
```

```
prompt> define_dft_partition part_2\  
-include {U3 U4}
```

The following example defines 2 design partitions for use in a DFT insertion. The design contains at the top level 4 hierarchical

cells. The cells *U1* and *U2* are instantiated from the *ref1* design reference and cells *U3* and *U4* are instantiated from design reference *ref2*:

```
prompt> define_dft_partition part_1\  
        -include {ref1}
```

```
prompt> define_dft_partition part_2\  
        -include {ref2}
```

SEE ALSO

- `current_dft_partition(2)`
- `insert_dft(2)`
- `preview_dft(2)`
- `remove_dft_partition(2)`
- `report_dft_partition(2)`
- `set_dft_signal(2)`
- `set_scan_compression_configuration(2)`
- `set_scan_configuration(2)`
- `set_scan_path(2)`

define_libcell_subset

Defines a subset of library cells to restrict the optimization of sequential and instantiated combinational cells.

SYNTAX

```
status define_libcell_subset  
  [-libcell_list lib_cells]  
  [-family_name name]
```

Data Types

<i>lib_cells</i>	list
<i>name</i>	string

ARGUMENTS

-libcell_list *lib_cells*

Specifies a list of library cells to be a family. These library cells must abide by the following rules: First, the library cells cannot belong to another family. Second, they must have the same functional identification. Third, they must be either sequential library cells or instantiated combinational cells.

-family_name *name*

Specifies a name for the subset of library cells.

DESCRIPTION

The **define_libcell_subset** command defines a subset of target library cells to be used for the **set_libcell_subset** command. After the library cells are grouped into a family, they are not available for use during the compile run. However, cells that are specified by the **set_libcell_subset** command can be optimized. A library cell can only belong to one family. Therefore, if you need to change the family of a library cell, the family should be removed first using the **remove_libcell_subset** command.

This command is supported only in topographical mode.

The exclusiveness restriction applies only to the new cells that are created or mapped during optimization. This restriction does not affect cells that are already mapped.

To remove a family, use the **remove_libcell_subset -family_name** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, **define_libcell_subset** groups logic library cells SDFLOP1 and SDFLOP2 into a family called special_flops. Next, **set_libcell_subset** sets the target family for instances u1 and u2 to the defined special cell family, special_flops, which consists of the cells SDFLOP1 and SDFLOP2.

```
prompt> define_libcell_subset -libcell_list "SDFLOP1 SDFLOP2" -family_name special_flops  
prompt> set_libcell_subset -object_list [get_cells u1 u2] -family_name special_flops
```

SEE ALSO

```
remove_libcell_subset(2)  
report_libcell_subset(2)  
set_libcell_subset(2)  
define_libcell_subset(2)
```

define_name_maps

Defines object name mapping for an application and design.

SYNTAX

```
status define_name_maps  
-application application_name  
-design_name design_name  
-columns column_list  
entries
```

Data Types

```
application_name  string  
design_name       string  
column_list      list  
entries          list_of_lists
```

ARGUMENTS

-application *application_name*

Specifies the targeting application name. Currently only *golden_upf* is supported. *application_name* and *design_name* jointly specify a unique in-memory name map instance for the current session.

-design_name *design_name*

Specifies the name of the design. The *application_name* and *design_name* values jointly specify a unique in-memory name map instance for the current session.

-columns *column_list*

Specifies the column names of the name map for the *application_name*.

entries

Specifies name mapping entries conforming to the definition of *column_list*.

DESCRIPTION

The **define_name_maps** command allows you or the tool to specify a list of object name mapping entries for a specified application. Usually a name map is associated with a particular design name. The *application_name* value is predefined by the tool.

The object name map can help in certain constraint applications where the default object query or even the rule-based query algorithms cannot succeed in finding the correct objects. The typical case is when trying to apply RTL constraints to the post-optimization netlist.

Note that it is the specific application that defines the content format and the usage model of its name map. To fully understand the functionality of a name map, see the related documentation for the specific application.

APPLICATION *golden_upf*

golden_upf is the application name assigned to the golden UPF flow. The command **define_name_maps** for *golden_upf* can appear in the golden UPF name mapping file, although other commands, such as **set_query_rules** can also exist in the golden UPF name mapping file. The *golden_upf* name map has four columns. They are *class*, *pattern*, *options*, and *names*.

Columns *class*, *pattern*, and *options* are used jointly to locate the entry in the map; the *names* column is the mapping result.

The following example illustrates the map contents and columns.

```
prompt> define_name_maps \
  -application golden_upf \
  -design TOP \
  -columns {class pattern options names} \
  {cell mid/inst/U3 {} {mid_inst/U4 mid_inst/U5}} \
  {cell mid/inst/bot+U5 {} {mid_inst/bot/U9}}
```

The preceding name map is for the application *golden_upf* and a design named TOP. Four columns are predefined for the application as follows:

- *class* specifies an object class name, such as cell or port. The supported class names are cell, port, pin, net, power_domain, power_switch, supply_net, supply_port, and supply_set.
- *pattern* specifies the object matching pattern string. This is one of the object name patterns from the user golden UPF file. The optional delimiter character '+' is used to indicate that a pattern is composed of a matching string (for example, "U5") and a scope (for example, scope "mid/inst/bot") for that matching string.
- *options* specifies a list of matching options for the map entry. It is derived from the user golden UPF file. The supported *options* is a list of the following elements: {*nocase* *transitive* *leaf* *noleaf* *in* *out* *inout*}. The *nocase* option specifies that the string in the *pattern* column should be treated case-insensitively during name lookups. See the documentation for the **find_objects** command for descriptions of the other options. Note that *leaf* and *noleaf* are mutually exclusive, and *in*, *out*, and *inout* are mutually exclusive. In the preceding example, no option is used.
- *names* is a list of object names where every name element is a full-path name relative to the top scope of the specified design. Column *names* is the name mapping target.

The preceding name map would result in the following query behavior under the *golden_upf* application, although in a real golden UPF flow, the queries are executed through the **load_upf** command:

```
prompt> find_objects . -pattern mid/inst/U3 -object_type inst
{mid_inst/U4 mid_inst/U5}

prompt> find_objects mid/inst/bot -pattern U5 -object_type inst
{mid_inst/bot/U9}
```

SEE ALSO

```
set_query_rules(2)
load_upf(2)
find_objects(2)
```

define_name_rules

Defines a set of name rules for designs.

SYNTAX

```
status define_name_rules
    name_rules
    [-max_length length]
    [-target_bus_naming_style bus_naming_style]
    [-allowed allowed_chars]
    [-restricted restricted_chars]
    [-first_restricted first_chars]
    [-last_restricted last_chars]
    [-reserved_words reserves]
    [-replacement_char char]
    [-remove_chars]
    [-equal_ports_nets]
    [-inout_ports_equal_nets]
    [-collapse_name_space]
    [-case_insensitive]
    [-special output_format]
    [-prefix prefix_name]
    [-map map_string]
    [-type object_type] [-reset]
    [-remove_internal_net_bus]
    [-remove_port_bus]
    [-check_bus_indexing]
    [-check_bus_indexing_use_type_info]
    [-rename_three_state_port_net]
    [-check_internal_net_name]
    [-remove_irregular_port_bus]
    [-remove_irregular_net_bus]
    [-flatten_multi_dimension_busses]
    [-preserve_struct_ports]
    [-dont_change_bus_members]
    [-dont_change_ports]
    [-add_dummy_nets]
    [-dummy_net_prefix dummy_nets_format]
    [-dir_inout_as_in]
```

Data Types

```
name_rules      string
length          integer
bus_naming_style string
allowed_chars   string
restricted_chars string
```

```

first_chars      string
last_chars       string
reserves         list
char             string
output_format    string
prefix_name      string
map_string       string
object_type      string
dummy_nets_format string

```

ARGUMENTS

name_rules

Specifies the name of the rules being defined. If named rules called *name_rules* do not exist, they are created.

-max_length *length*

Specifies the maximum length of a name, which must be 8 or more characters, except that a value of 0 resets the maximum length to its default (any length).

-target_bus_naming_style *bus_naming_style*

Specifies the target bus naming style for all object types, to which the current bus naming style is to be changed. For this option to succeed, set the **bus_naming_style** variable with the current bus naming style. For an example, see the section entitled "Changing the Bus Naming Style." By default, the target bus naming style is set to be the same as the current bus naming style. If the target bus naming style is different from the current bus naming style, after the rule is applied through the **change_names** command, the bus naming style becomes the target bus naming style, but this does not occur automatically. If more **change_names** commands need to be issued later, be careful that the bus naming style is in transition from one rule to another rule and needs to be set manually to reflect the real bus style inside the .db file during the transition.

-allowed *allowed_chars*

Specifies the set of characters allowed in names, which must be 10 or more characters. The format of the *allowed_chars* string is provided in the "DESCRIPTION" section. By default, any printable character is allowed in a name.

-restricted *restricted_chars*

Specifies the set of characters not allowed in names. The format of the *chars* string is provided in the "DESCRIPTION" section. By default, any printable character is allowed in a name.

-first_restricted *first_chars*

Specifies a set of characters that are not allowed as the first character in a name. The format of the *chars* string is provided in the "DESCRIPTION" section. By default, any printable character is allowed as the first character of a name.

-last_restricted *last_chars*

Specifies a set of characters not allowed as the last character in a name. The format of the *chars* string is described in the "DESCRIPTION" section. By default, any printable character is allowed as the last character of a name.

-reserved_words *reserves*

Specifies a list of words that cannot be used as a name. The *reserves* value contains words that are considered reserved in your target system. By default, there are no reserved words.

-replacement_char *char*

Specifies the character used to replace characters not allowed in names, as specified by the **-allowed** or **-restricted** option. By default, the replacement character is the underscore character (`_`).

-remove_chars

Specifies that characters not allowed in names, as specified by the **-allowed** or **-restricted** option, are removed rather than replaced. By default, illegal characters are replaced by the character specified with **-replacement_char**.

-equal_ports_nets

Specifies that nets connected to non-inout ports must have the same name as their connecting port. If a net is connected to more than one port, the net's name is not changed and a warning is issued. By default, nets are not required to match the names of the non-inout ports to which they are connected.

-inout_ports_equal_nets

Specifies that nets connected to inout ports must have the same name as their connecting port. If a net is connected to more than one port, the net's name is not changed and a warning is issued. By default, nets are not required to match the names of the inout ports to which they are connected.

-collapse_name_space

Specifies that the name space of ports, cells, and nets is the same. No port, cell, or net in a design can have the same name. By default, ports, cells, and nets each have their own name space.

-case_insensitive

Specifies that the case of characters is not significant when comparing names. For example, the name *example* (lowercase) is equivalent to *EXAMPLE* (uppercase). If two names are equivalent within a name space, one of the names must be changed. By default, the case of a name is significant.

-special_output_format

Specifies to use special rules pertaining to a specific target system when changing names. By default, no special rules apply. The possible values for *output_format* are as follows:

```
verilog
vhdl
sge
vhdl93
sge_vhdl
siff
```

-prefix_prefix_name

Specifies a prefix used when changing a name. By default, the prefixes are **P** for ports, **U** for cells, and **N** for nets.

Use this option only when the **change_names** command needs to create a completely new name to ensure the name's uniqueness. See Step 3 of the "Naming Rules" section for more information.

-map_map_string

Provides a way to change objects in addition to the previously specified name rules. This option specifies the name mapping and replacement rules as defined in the *map_string*. See "Mapping Rules" under the "Naming Rules" section for details.

-type_object_type

Specifies that the rules being defined apply only to the given type of objects. Allowed values for *object_type* are **port**, **cell**, or **net**. By default, the defined rules apply to all object types.

-reset

Resets all rules to their default values. When all rules are set to the defaults, no restrictions are imposed on the names.

-remove_internal_net_bus

Bit-blasts all internal net buses.

-remove_port_bus

Bit-blasts all port buses. Net buses that connect to ports are also bit-blasted. This option overrides the **-preserve_struct_ports** option, setting its value to the default value.

-check_bus_indexing

Checks the indices of buses and bit-blasts the incomplete buses.

-check_bus_indexing_use_type_info

Checks bus indexing based on the bus type information and bit-blast for incomplete port buses.

-rename_three_state_port_net

Renames three-state port nets so they have different names from the ports on the inout port connections.

-check_internal_net_name

Checks and renames nets that have the same name as some ports, but the nets are not connected to these ports.

-remove_irregular_port_bus

Bit-blasts any port bus that contains irregular members.

-remove_irregular_net_bus

Bit-blasts any net bus that contains irregular members.

-flatten_multi_dimension_busses

Flattens multi-dimension ports, net buses, and arrays so writing out is easier.

-preserve_struct_ports

Specifies to avoid bit-blasted ports inferred from SystemVerilog structures or VHDL records.

-dont_change_bus_members

Specifies not to change the elements that are in either a port bus or a net bus. This option overrides the previous four options (**-remove_irregular_port_bus**, **-remove_irregular_net_bus**, **-flatten_multi_dimension_busses**, and **-preserve_struct_ports**). If this option is set, these four options are reset to the default value of **false**. In most cases, this option should remain **false**.

-dont_change_ports

Specifies not to change elements that are set by the **-type** option to **port**.

-add_dummy_nets

Adds dummy nets for unconnected pins. The **-dummy_net_prefix** option specifies the format to name dummy nets. The format defaults to `SYNOPSIS_UNCONNECTED_%d`.

-dummy_net_prefix *dummy_nets_format*

Sets the prefix for the dummy net naming convention. The *dummy_nets_format* value specifies the format to name the dummy nets.

-dir_inout_as_in

Instructs the tool to treat all instances with the INOUT direction as if they were set to IN. The default behavior is to treat them as direction OUT.

DESCRIPTION

The **define_name_rules** command defines a set of rules for naming design objects. Name rules are used by the **change_names** and **report_names** commands. The **report_name_rules** command displays a listing of the name rules currently defined in the shell.

Name rules can be defined in multiple calls to the **define_name_rules** command. For an example of multiple calls, see the "EXAMPLES" section.

The **-type** option enables you to define rules that apply to a specific object type, such as port, cell, or net). Each call to **define_name_rules** is additive or overrides previous calls for a specific name rules.

Naming Rules

This section describes the effects of specific options when defining name rules with **define_name_rules**.

Maximum Length Rules

The maximum number of characters in a name can be restricted with the **-max_length** option.

Names shorter than the specified maximum length remain unchanged. Names longer than the specified maximum length are modified so that their lengths are less than or equal to the maximum.

Fixed names are guaranteed to be unique within the design. The following steps are applied in succession to fix a name to meet the uniqueness criteria:

- Step 1
Truncation - Names are truncated to meet the maximum length. The truncated name is accepted if it is unique within the design. For example, if the name is THIS_NAME_IS_TOO_LONG_TO_BE_ACCEPTABLE and the maximum number of characters is 16, the resulting name is THIS_NAME_IS_TOO.
- Step 2
Truncation with index - Names are truncated. The truncated name is appended with an index in an attempt to make it unique. If a unique name is found, it is accepted. If the name from the previous example, THIS_NAME_IS_TOO, is not unique, this the tool tries THIS_NAME_IS_T1, THIS_NAME_IS_T2, and continues up to, THIS_NAME_IS_T99. The first unique name is accepted.
- Step 3
Basic names - If both previous steps fail, the original name is deleted and names of the form <prefix><index> are generated until a unique name is found. The <prefix> is defined using the **-prefix** option. Examples are N1, N2, and so on.

If all three steps fail to generate a unique name, the original name is retained and a warning message is issued.

Character Restriction Rules

Specify characters that make up names with the **-allowed**, **-restricted**, **-first_restricted**, and **-last_restricted** options.

To specify the character set allowed, use **-allowed** or **-restricted**. Either of these options overrides the previous values of **-allowed**, **-restricted**, **-first_restricted**, and **-last_restricted**.

In conjunction with **-allowed** or **-restricted**, use **-first_restricted** and **-last_restricted** to further restrict the available characters

for the first and last character of a name. The **-first_restricted** and **-last_restricted** options depend on **-allowed** and **-restricted**, and modify the set of available characters for their special cases. The set of characters available for first and last characters cannot exceed that defined for all other characters.

The *allowed_chars*, *restricted_chars*, *first_chars*, and *last_chars* strings contain characters that can be used (they are **-allowed**), or cannot be used (they are **-restricted**) in design object names. The dash (-) character indicates a range of ASCII characters. To include a literal dash character in a *chars* parameter, precede it with 2 backslashes (\-). Blank spaces are disregarded. A minimum of 10 characters must be allowed or an error is reported. The order of the characters in the *chars* string parameters is not significant.

The following example specifies a character set consisting of uppercase characters and an underscore.

```
prompt> define_name_rules -allowed "A-Z _"
```

In the following example, some punctuation characters are restricted. The dash (-) character is restricted with the character sequence \-. The double quotation marks (") are restricted with the sequence \". First characters cannot be lowercase.

```
prompt> define_name_rules -restricted "!@#%&*()\\-\" \"
      -first_restricted "a-z"
```

The following example attempts to restrict the character set to fewer than 10 characters. An error is reported.

```
prompt> define_name_rules -allowed "ABCDE"
```

If a design object name contains a character that is restricted for its object type, the following steps are taken to fix the name:

- Step 1
Remove the character - If **-remove_chars** is specified in the name rules, the offending character is removed.
- Step 2
Change the case - The case of characters is changed to meet character restrictions. The change is accepted if the new character is allowed. For example, if names are restricted to uppercase characters, the name `sum_port` is changed to `SUM_PORT`.
- Step 3
Replace the character - A restricted character is changed to the value of **-replacement_char**. For example, if punctuation characters are not allowed and the replacement character is an underscore, the name `U$1` is changed to `U_1`.
- Step 4
Uniquify the name - If the name created in either Step 1 or Step 2 is not unique, an index is appended to the new name.

Reserved Word Rules

Prohibit specific names from becoming design object names by using the **-reserved_words**. The parameter to this option is a list of names that are not allowed as design object names.

The following example uses **-reserved_words**. The words `DESIGN`, `MODULE`, `START`, and `END` are designated as reserved. The case of the names in this list is significant.

```
prompt> define_name_rules \
      -reserved_words {"DESIGN", "MODULE", "START", "END"}
```

Case Insensitive Rules

If **-case_insensitive** is specified, the case of the characters in a name is not significant when comparing names.

If **-case_insensitive** is specified, the case of characters in reserved words is not significant. For example, if **-case_insensitive** is used and the string `MODULE` is specified as a reserved word, the name `module` is also considered a reserved word and changed.

Port and Net Rules

Make the name of a net connected to a port equal to the name of that port with **-equal_ports_nets**. If this substitution causes a conflict, no names are changed and a warning message is issued. If a net is connected to more than one port, no names are changed and a warning message is issued.

Name Space Rules

By default, name spaces of ports, cells, and nets are separate. Names of ports and cells must be unique within a design. A port, a cell, and a net within a design can share the same name.

Name spaces for ports, cells, and nets can be shared using the **-collapse_name_space** option. Ports, cells, and nets within a design must be unique across all objects. Conflicting names are modified by appending digits to their name.

If the **-collapse_name_space** option is specified with **-equal_ports_nets**, **-equal_ports_nets** has priority over **-collapse_name_space**. So you may still have a net that is named same as the connected port.

Type-Specific Rules

Name rules can be tailored for ports, cells, and nets object types with the **-type** option. The options to **define_name_rules** that can be specified by object type are as follows:

- max_length**
- allowed**
- restricted**
- first_restricted**
- last_restricted**
- replacement_char**
- remove_chars**
- prefix**

To specify the maximum length restriction on ports different than the maximum length for cells, make multiple calls to **define_name_rules**.

In the following example, the maximum length of ports is set to 12. The maximum length of cells is set to 8. The maximum length of nets is unrestricted.

```
prompt> define_name_rules EXAMPLE -max_length 12 -type port
```

```
prompt> define_name_rules EXAMPLE -max_length 8 -type cell
```

Special Rules

Define name rules that are for a specific target system and cannot be modeled with the general **define_name_rules** options. The systems supported by the **-special** option are *sge*, *verilog*, *vhdl*, *vhdl93*, *sge_vhdl*, and *siff*.

Specifying **-special verilog** adds one rule to make names conform to the rules of Verilog.

The difference between **-special verilog** and **-rules verilog** is as follows:

- For **-special verilog**, **verilog** means the output format for a specific target system. The following rules will be specified automatically along with **[-special verilog]**:

- collapse_name_space**
- equal_ports_nets**
- inout_ports_equal_nets**
- remove_irregular_port_bus**
- remove_irregular_net_bus**

- For **-rules verilog**, **verilog** means the name of the rule. The verilog rule is a predefined name rule in **change_names** and the recommended rule. The following rules are already defined in verilog rule:

```

-special verilog
-target_bus_naming_style {%s[%d]}
-flatten_multi_dimension_busses
-check_internal_net_name
-check_bus_indexing
...

```

You can add other rules to the verilog rule set.

- Specifying **-special vhd** adds one rule to make names conform to the rules of VHDL. This option does not allow consecutive underscores in a name.
- Specifying **-special vhd93** adds one rule to make names conform to the rule of VHDL93. This option adds support for an extended identifier. An extended identifier is a sequence of characters that are defined by the VHDL93 character set and are written between two backslashes (\).
- Specifying **-special sge** adds one rule to make names conform to the rules of the SGE system. This option does not allow net names that start with the characters BBBB_ and NNNN_. These names are reserved by the SGE system.
- Specifying **-special sge_vhdl** is a combination of the previous two rules. The **change_names** command enforces **-collapse_name_space** and **-equal_ports_nets**, but because these two rules conflict with each other, the **sge_vhdl** rule applies under the following conditions:
 - The **-collapse_name_space** rule first checks the object class type. If the object is a port, it checks to see if the name is used by any port names. Since the ports are processed first, there are no collisions with the other name spaces. If the object is a cell or a design, it verifies that the name is not used previously by any port. If the object is a net, it checks the port, and cell name spaces. If there is a name conflict, it changes the name so that it is unique. See the **-collapse_name_space** rule description for details.
 - The **-equal_ports_nets** rule is applied to a scalar net or a net object whose owning bus is not of a single width. See the **-equal_ports_nets** rule description for details.
- Specifying **-special siff** adds one rule to make names conform to the rules for the "Synopsys Integrator For Falcon Framework."

All of the special rules previously given share the same name rules in addition to their own name rules:

- The names of bused ports or nets follow the *bus_naming_style*, except that *sge_vhdl* always uses "%s(%d)" instead of *bus_naming_style*.
- The names of bused ports or nets owned by the same bus have the same base name. The bus index order follows the index part defined in the owning bus.
- If a net does not belong to a busbag and it contains a separator character defined in the **bus_naming_style** variable, this separator character changes to _ (underscore) and the ending separator is removed. For example, the net name *sample[24]/[3]* changes to the name *sample_24_3*.

Specifying **-special** does not create all of the rules necessary for VHDL, SGE, or SIFF compliance. Only the non-general rules previously described are created. To make **define_name_rules** to constrain names to fully meet the requirements of SGE and VHDL, use the other options in conjunction with **-special**.

To avoid conflict, do not merge special name rules with name rules that you defined. When defined name rules conflict, **change_names** could generate names that are not compliant to your name rule. When there is a conflict among name rules, define each one of those name rules using a different name in **define_name_rules** and apply each of them using **change_names**. The last rule applied generates names that override names generated by previous rules.

Mapping Rules

The **-map** option specifies the name mapping and replacement rules as defined in the *map_string*. The mapping string has the following format:

```
{{"pattern","replacement"} [{"pattern","replacement"}]* }
```

The mapping string is grouped into pairs. There can be any number of pairs in a mapping string. Each pair is enclosed by {} (curly braces) and separated by a , (comma). The **change_names** and **report_names** commands apply all of them according to the order specified in the *map_string*.

The first member of each pair is the *pattern* string, which is used to match against the object name. The second member is the *replacement* string, which is used to replace the matched portion of the object name. Both the *pattern* and *replacement* string must begin and end with " (double quotation marks). There is a , (comma) between *pattern* and *replacement* string.

The following example has the first occurrence of any portion of an object name that contains **_reg** replaced by an empty string and the first occurrence of any portion of a object name that contains **A** replaced by **a** in the new name.

```
prompt> define_name_rules my_rule -map {"_reg", ""}, {"A", "a"}
```

The pattern string also provides limited regular expression capability. The following 4 special characters are supported to make the mapping strings more specific:

- The \$ (dollar sign) indicates the end of string.
- The ^ (caret) indicates the beginning of string.
- The * (asterisk) indicates a wildcard matching 0 or more characters.
- The ? (question mark) indicates a wildcard matching 1 character.

The following example defines a rule that any name ending with **_reg** is replaced by **in** at the end of the string:

```
prompt> define_name_rules my_rule0 -map {"_reg$", "in"}
```

The following example defines a rule that any name beginning with **_reg** is replaced by **in** at the beginning of the string:

```
prompt> define_name_rules my_rule1 -map {"^_reg", "in"}
```

The following example defines a rule that the first occurrence of any substring of a name that begins with **_r** and ends with **g** is replaced by the **in** string:

```
prompt> define_name_rules my_rule2 -map {"_r*g", "in"}
```

The following example defines a rule that the first occurrence of any substring of a name that begins with **_r** followed any one character, then ending with **g** is replaced by **in**:

```
prompt> define_name_rules my_rule3 -map {"_r?g", "in"}
```

Once a mapping rule is defined, you can only use **-reset** to reset the mapping rule. You cannot redefine the same matching pattern within the same rule. For example, assume you define the following rule:

```
prompt> define_name_rules my_rule4 -map {"A", "X"}
```

If you then decide to replace all occurrences of **A** with **Z**, you must first execute a reset and then redefine the name rules as follows:

```
prompt> define_name_rules my_rule4 -reset
```

```
prompt> define_name_rules my_rule4 -map {"A", "Z"}
```

To insert escape characters into names, you can also use the **-map** option. This is useful prior to generating reports, if the report output is processed by a program that requires certain characters to be escaped. For example, if names contain the / (slash) character, the tool cannot differentiate between a / used in a name and a / that denotes a change in hierarchy. Defining a rule with the following command replaces any / in a name with \ (backslash slash).

```
prompt> define_name_rules -map {"/", "\\"}
```

The name **HAS/SLASH** inside hierarchy **A** writes out as **A/HAS\SLASH**.

Default Rules Values

Once defined, you can reset name rules to their default values with the **-reset** option. Name rules with all default values impose no restrictions on design object names.

Default values for specific rules are shown below. A set of name rules called **ALL_DEFAULTS** is defined with no options. The **report_name_rules** command displays the default value of each specific rule.

```
prompt> define_name_rules ALL_DEFAULTS
```

```
prompt> report_name_rules ALL_DEFAULTS
```

```
*****
Report : name_rules
Name Rules : ALL_DEFAULTS
Version: v3.0
Date   : Fri Sep 27 09:36:52 1991
*****
```

```
Rules Name: ALL_DEFAULTS
Equal port and net names: false
Collapse name space: false
Case insensitive: false
Special rules: none
Reserved words: none
```

```
      Max Repl Rem
Rules Type Len Char Chrs Prefix Allowed Chars
-----
Port Rules none '_' no P    No restrictions
Cell Rules none '_' no U    No restrictions
Net Rules  none '_' no N    No restrictions
```

Many rules can be reset individually by specifying empty values for the option. The following table shows how to specify the default value for these fields. Options not listed in the table must be reset with the **-reset** option.

Option	Default Value
-----	-----
-max_length	0
-restricted	""
-allowed	""
-first_restricted	""
-last_restricted	""
-reserved_words	{}
-replacement_char	""
-prefix	""

Applying Name Rules

When names are changed using the **change_names** or **report_names** command, name rules are applied sequentially even though they are specified concurrently in the **define_name_rules** command. Name rules are applied in order, so rules applied later might overwrite previous names rules and there will be cases where names after **change_names** or **report_names** do not conform to all rules specified.

This section describes the order in which name rules are applied, and the results you can expect.

The name rules are applied in the following order:

1. Map rule: rules that are applied to an object name string. The rule in this category is **-map_rule**.
2. Character rules: rules that are applied to object name characters. Rules in this category are **-allowed**, **-restricted**, **-first_restricted**, and **-last_restricted**. To support the character rules, use **-remove_char** and **-replacement_chars**.
3. Design rules: rules that are applied to object names in terms of whole design naming methodology. Rules in this category are **-equal_ports_nets** (for net objects only), **-special**, **-max_length**, and **-reserve_words**.
4. Meta rules: rules that extend beyond one of the previous single categories. Rules in this category are **-collapse_name_space**, **-case_insensitive**, **-prefix**, and **-type**.

Since rules applied earlier might be overwritten by later rules, apply the "must have" rule last. The name rules are applied to object names according to the order shown above. That is, the string rules are applied first, then the character rules, and the design rules are last. Meta rules are used for supporting those rules during name changes. Rules within the same category are applied following the same order as the prior list. Because later rules might override previous ones, to preserve the priority it is better to use those rules that cause conflict separately. Among these rules, use mapping primarily.

During name changes, the objects are categorized into three types: port, cell, and net. Port objects are changed first, then cell objects, then net objects. When there is a name conflict among a port name, a cell name, and a net name, the port name has highest priority (it is not changed). The net and port names must be changed to unique names.

Changing the Bus Naming Style

When **define_name_rules** changes the bus naming style, use the **bus_naming_style** variable to identify the bus naming style used by the current database.

For example, the following script changes the bus naming style from [] to < >:

```
prompt> define_name_rules change_bus_naming_style \
    -target_bus_naming_style "%s<%d>" \
    bus_naming_style="%s[%d]"

prompt> change_names -rule change_bus_naming_style -hierarchy
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines a name rule called *SIMPLE* that requires design names in uppercase. The *SIMPLE* name rule is used as the default name rule by **change_names**.

```
prompt> define_name_rules SIMPLE -allowed "A-Z_"

prompt> default_name_rules = SIMPLE

prompt> change_names
Information: Using name rules 'SIMPLE'.
Information: 153 names changed in design 'MY_DESIGN'.
```

The following example defines name rules called *MY_RULES* using multiple calls to **define_name_rules**. The length of names is restricted to 16 characters. The example defines the allowed character set and a replacement character:

```
prompt> define_name_rules MY_RULES -max_length 16
```

```
prompt> define_name_rules MY_RULES -replacement_char "X"
```

```
prompt> define_name_rules MY_RULES -allowed "A-Z_"
```

In the following example, multiple calls to **define_name_rules** use most of the options. The **report_name_rules** command displays the effect of these calls.

```
prompt> define_name_rules SMORG -collapse_name_space
```

```
prompt> define_name_rules SMORG -case_insensitive
```

```
prompt> define_name_rules SMORG -reserved_words {IN,OUT,INOUT}
```

```
prompt> define_name_rules SMORG -case_insensitive
```

```
prompt> define_name_rules SMORG -type port -max_length 16 \
    -allowed "A-Z a-z 0-9_*" \
    -replacement_char "***"
```

```
prompt> define_name_rules SMORG -type cell -prefix "CELL" \
    -restrict "()[]{}"
```

```
prompt> define_name_rules SMORG -type net -allowed "A-Za-z_*" \
    -first_restrict "0-9" -last_restrict "0-9" \
    -replacement_char "***"
```

```
prompt> report_name_rules SMORG
```

```
*****
Report : name_rules
Name Rules : SMORG
Version: v3.0
Date   : Fri Sep 27 11:00:09 1991
*****
```

```
Rules Name: SMORG
  Equal port and net names: false
  Collapse name space: true
  Case insensitive: true
  Special rules: none
  Reserved words: {IN, OUT, INOUT}
```

```
      Max Repl Rem
Rules Type Len Char Chrs Prefix Allowed Chars
-----
Port Rules 16 '*' no P    Use "A-Z a-z 0-9_*"
Cell Rules none '_' no CELL Don't use "()[]{}"
Net Rules none '*' no N    Use "A-Za-z_*"
                        First: Don't use "0-9"
                        Last: Don't use "0-9"
```

SEE ALSO

change_names(2)
report_names(2)
report_name_rules(2)
bus_naming_style(3)

define_power_model

Defines a power model which describes the power intent of a reference library cell.

SYNTAX

```
status define_power_model  
    power_model_name  
    [-for library_cells]  
    {UPF_commands}
```

Data Types

```
power_model_name    string  
library_cells       list  
UPF_commands        command list
```

ARGUMENTS

power_model_name

Specifies the name of the power model to be defined. The name should be a simple (non-hierarchical) name.

The name is scope-based, i.e., the power model cannot be defined if there is another power model with the same name in the scope where the `define_power_model` command is executed.

-for library_cells

Specifies a collection of library cells that this power model can be applied to.

If this option is not specified, the power model can then be applied to any library cell.

DESCRIPTION

The `define_power_model` command defines a power model containing other UPF commands contained within curly braces. A power model is used to define the power intent of a model and shall be used in conjunction with one or more model representations. The opening and closing curly braces encapsulating the UPF commands that describe the power model must be present in the same UPF file.

A power model can be referenced by its simple name from anywhere in a power intent description. It shall be an error to have two power models with the same name and at the same scope.

A power model can be applied to specific instances using `apply_power_model`.

A power model that is not referenced by an `apply_power_model` command does not have any impact on the power intent of the design.

The `define_power_model` command provides a protected environment to the power model. Any Tcl variable used in the power model is not affected by the global variables defined outside the `define_power_model` command. Also, any Tcl variable defined inside the power model will not affect the environment outside the power model.

A power model can be parameterized by creating parameters using the `add_parameter` command. These parameters are represented as special Tcl variables which can be referenced within the power model. The parameters can be overridden with `-parameters` option of the `apply_power_model` command.

A power model cannot have a nested definition of power models, i.e. a nested `define_power_model` command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a power model for two library cells. Notice that there is a parameter for the top power domain name in the power model.

```
prompt> define_power_model MODEL -for {lib_cell_A lib_cell_B} {  
  add_parameter DOMAIN -default domain -description "top power domain in model"  
  create_supply_net VDD  
  create_supply_net VSS  
  create_supply_set SS -function {power VDD} -function {ground VSS}  
  create_power_domain PD_${DOMAIN} -supply {primary SS} -include_scope  
}
```

SEE ALSO

`apply_power_model(2)`
`add_parameter(2)`
`report_power_model(2)`
`upf_power_model_library(3)`
`upf_power_model_search_path(3)`

define_proc_attributes

Defines attributes of a Tcl procedure, including an information string for help, a command group, a set of argument descriptions for help, and so on. The command returns the empty string.

SYNTAX

```
string define_proc_attributes
    proc_name
    [-info info_text]
    [-define_args arg_defs]
    [-define_arg_groups group_defs]
    [-command_group group_name]
    [-return type_name]
    [-hide_body]
    [-hidden]
    [-dont_abbrev]
    [-permanent]
    [-deprecated]
    [-obsolete]
```

Data Types

```
proc_name    string
info_text    string
arg_defs     list
group_defs   list
group_name   string
type_name    string
```

ARGUMENTS

proc_name

Specifies the name of the existing procedure.

-info *info_text*

Provides a help string for the procedure. This is printed by the **help** command when you request help for the procedure. If you do not specify *info_text*, the default is "Procedure".

-define_args *arg_defs*

Defines each possible procedure argument for use with **help -verbose**. This is a list of lists where each list element defines one argument.

-define_arg_groups group_defs

Defines argument checking groups. These groups are checked for you in `parse_proc_arguments`. each list element defines one group

-command_group group_name

Defines the command group for the procedure. By default, procedures are placed in the "Procedures" command group.

-return type_name

Specifies the type of value returned by this proc. Any value may be specified. Some applications use this information for automatically generated dialogs etc. Please see the `type_name` option attribute described below.

-hide_body

Hides the body of the procedure from **info body**.

-hidden

Hides the procedure from **help** and **info proc**.

-dont_abbrev

Specifies that the procedure can never be abbreviated. By default, procedures can be abbreviated, subject to the value of the `sh_command_abbrev_mode` variable.

-permanent

Defines the procedure as permanent. You cannot modify permanent procedures in any way, so use this option carefully.

-deprecated

Defines the procedure as deprecated i.e. it should no longer be called but is still available.

-obsolete

Defines the procedure as obsolete i.e. it used to exist but can no longer be called.

DESCRIPTION

The **define_proc_attributes** command associates attributes with a Tcl procedure. These attributes are used to define help for the procedure, locate it in a particular command group, and protect it.

When a procedure is created with the **proc** command, it is placed in the Procedures command group. There is no help text for its arguments. You can view the body of the procedure with **info body**, and you can modify the procedure and its attributes. The **define_proc_attributes** command allows you to change these aspects of a procedure.

Note that the arguments to Tcl procedures are all named, positional arguments. They can be programmed with default values, and there can be optional arguments by using the special argument name *args*. The **define_proc_attributes** command does not relate the information that you enter for argument definitions with **-define_args** to the actual argument names. If you are describing anything other than positional arguments, it is expected that you are also using **parse_proc_arguments** to validate and extract your arguments.

The *info_text* is displayed when you use the **help** command on the procedure.

Use **-define_args** to define help text and constraints for individual arguments. This makes the help text for the procedure look like the help for an application command. The value for **-define_args** is a list of lists. Each element has the following format:

arg_name option_help value_help data_type attributes

The elements specify the following information:

- *arg_name* is the name of the argument.
- *option_help* is a short description of the argument.
- *value_help* is the argument name for positional arguments, or a one word description for dash options. It has no meaning for a Boolean option.
- *data_type* is optional and is used for option validation. The *data_type* can be any of: string, list, boolean, int, float, or one_of_string. The default is string.
- *attributes* is optional and is used for option validation. The *attributes* is a list that can have any of the following entries:
 - "required" - This argument must be specified. This attribute is mutually exclusive with optional.
 - "optional" - Specifying this argument is optional. This attribute is mutually exclusive with "required."
 - "value_help" - Indicates that the valid values for a one_of_string argument should be listed whenever argument help is shown.
 - "values {<list of allowable values>}" - If the argument type is one_of_string, you must specify the "values" attribute.
 - "type_name <name>" - Give a descriptive name to the type that this argument supports. Some applications may use this information to provide features for automatically generated dialogs, etc. Please see product documentation for details. This attribute is not supported on boolean options.
 - "merge_duplicates" - When this option appears more than once in a command, its values are concatenated into a list of values. The default behavior is that the right-most value for the option specified is used.
 - "remainder" - Specifies that any additional positional arguments should be returned in this option. This option is only valid for string option types, and by default the option is optional. You can require at least one item to be specified by also including the required option.
 - "deprecated" - Specifying this option is deprecated i.e. it should no longer be used but is still available. A warning will be output if this option is specified. This attribute cannot be combined with obsolete or required.
 - "obsolete" - Specifying this option is obsolete i.e. it used to exist but can no longer be used. A warning will be output if this option is specified. This attribute cannot be combined with deprecated or required.
 - "min_value" <value> - Specify the minimum value for this option. This attribute is only valid for integer and float types.
 - "max_value" <value> - Specify the maximum value for this option. This attribute is only valid for integer and float types.
 - "default" <value> - Specify the default value for this option. This attribute is only valid for string, integer and float option types. If the user does not specify this option when invoking the command this default value will be automatically passed to the associated tcl procedure.

The default for *attributes* is "required."

Use the **-define_arg_groups** to define argument checking groups. The format of this option is a list where each element in the list defines an option group. Each element has the following format:

```
{<type> {<opt1> <opt2> ...} [<attributes>]}
```

The types of groups are

- *"exclusive"* Only one option in an exclusive group is allowed. All the options in the group must have the same required/optional status. This group can contain any number of options.

- *"together"* If the first option in the group is specified then the second argument is also required. This type of group can contain at most two options.
- *\fi"related"* These options are related to each other. An automatic dialog builder for this command may try to group these options together.

The supported attributes are:

- *{"label" <text>}* An optional label text to identify this group. The label may be used by an application to automatically build a grouping in a generated dialog.
- *"bidirectional"* This is only valid for a together group. It means that both options must be specified together.

Change the command group of the procedure using the **-command_group** command. Protect the contents of the procedure from being viewed by using **-hide_body**. Prevent further modifications to the procedure by using **-permanent**. Prevent abbreviation of the procedure by using **-dont_abbrev**.

EXAMPLES

The following procedure adds two numbers together and returns the sum. For demonstration purposes, unused arguments are defined.

```
prompt> proc plus {a b} { return [expr $a + $b]}

prompt> define_proc_attributes plus -info "Add two numbers" \
? -define_args {
  {a "first addend" a string required}
  {b "second addend" b string required}
  {"-verbose" "issue a message" "" boolean optional}}

prompt> help -verbose plus
Usage: plus  # Add two numbers
[-verbose]  (issue a message)
a           (first addend)
b           (second addend)

prompt> plus 5 6
11
```

In the following example, the `argHandler` procedure accepts an optional argument of each type supported by **define_proc_attributes**, then displays the options and values received. Note the only one of `-Int`, `-Float`, or `-Bool` may be specified to the command:

```
proc argHandler {args} {
  parse_proc_arguments -args $args results
  foreach argname [array names results] {
    echo $argname = $results($argname)
  }
}

define_proc_attributes argHandler \
-info "Arguments processor" \
-define_args {
  {-Oos "oos help" AnOos one_of_string
   {required value_help {values {a b}}}}
```

```
{-Int "int help"  AnInt int  optional}
{-Float "float help"  AFloat float  optional}
{-Bool "bool help"  ""  boolean optional}
{-String "string help" AString string optional}
{-List "list help"  AList list  optional}
{-IDup "int dup help" AIDup int  {optional merge_duplicates}}
} \
-define_arg_groups {
  {exclusive {-Int -Float -Bool}}
}
```

SEE ALSO

help(2)
info(2)
parse_proc_arguments(2)
proc(2)
sh_command_abbrev_mode(3)

define_routing_rule

Defines design-specific, nondefault routing rules that are stored in the design database.

SYNTAX

```
status define_routing_rule
  rule_name
  [-reference_rule_name ref_rule_name
  | -default_reference_rule]
  [-widths {layer_name_and_width_pairs}]
  [-spacings {layer_name_and_spacing_pairs}]
  [-spacing_weights {layer_name_and_weight_pairs}]
  [-spacing_weight_levels {layer_name_and_weight_level_pairs}]
  [-spacing_length_thresholds {layer_name_and_length_threshold_pairs}]
  [-shield]
  [-shield_widths {layer_name_and_shield_width_pairs}]
  [-shield_spacings {layer_name_and_shield_spacing_pairs}]
  [-snap_to_track]
  [-via_cuts {via_definition_list}
  | -cuts {cut_definition_list}]
  [-taper_level tapering_level]
  [-taper_distance tapering_distance]
  [-driver_taper_distance driver_tapering_distance]
  [-multiplier_width layer_width]
  [-multiplier_spacing layer_spacing]
  [-double_pattern_mask_constraints layer_and_constraint_pairs]
  [-via_spacings via_layer_name_pairs_and_spacing]
  [-rdl_taper_distance rdl_tapering_distance]
  [-rdl_taper_widths {layer_name_and_rdl_tapering_width_pairs}]
  [-taper_over_pin_layers number_of_layers]
  [-taper_under_pin_layers number_of_layers]
  [-single_side_spacing]
```

Data Types

<i>rule_name</i>	string
<i>ref_rule_name</i>	string
<i>layer_name_and_width_pairs</i>	list
<i>layer_name_and_spacing_pairs</i>	list
<i>layer_name_and_weight_pairs</i>	list
<i>layer_name_and_weight_level_pairs</i>	list
<i>layer_name_and_length_threshold_pairs</i>	list
<i>layer_name_and_shield_width_pairs</i>	list
<i>layer_name_and_shield_spacing_pairs</i>	list
<i>via_definition_list</i>	list
<i>cut_definition_list</i>	list
<i>tapering_level</i>	integer

<i>tapering_distance</i>	float
<i>driver_tapering_distance</i>	float
<i>layer_width</i>	float
<i>layer_spacing</i>	float
<i>layer_and_constraint_pairs</i>	list
<i>via_layer_name_pairs_and_spacing</i>	list
<i>rdl_tapering_distance</i>	float
<i>layer_name_and_rdl_tapering_width_pairs</i>	list
<i>number_of_layers</i>	integer

ARGUMENTS

rule_name

Specifies the name for the new nondefault routing rule.

This is a required argument.

-reference_rule_name *ref_rule_name*

Specifies the name of the reference (source) rule to use.

The **-reference_rule_name** and **-default_reference_rule** options are mutually exclusive; you can specify only one. If you do not specify either option, the command uses the default routing rule as the reference rule.

-default_reference_rule

Specifies that the default routing rule is used as the reference rule.

The **-reference_rule_name** and **-default_reference_rule** options are mutually exclusive; you can specify only one. If you do not specify either option, the command uses the default routing rule as the reference rule.

-widths {*layer_name_and_width_pairs*}

Specifies the routing width for each named routing layer. Each entry in the list contains a routing layer name and its width in microns, separated by a space. The width portion of the pair is a floating point number. You can specify only a single width per layer.

If you do not specify the width for a layer, the tool sets the width based on the reference rule.

-spacings {*layer_name_and_spacing_pairs*}

Specifies the minimum different-net spacing allowed between wires for each named routing layer.

For Zroute, each entry in the list contains a routing layer name and a list of one or more spacing values in microns. The spacing values are floating point numbers. If you specify more than one spacing value, you must also use the **-spacing_weights** or **-spacing_weight_levels** option. The spacing values in the **-spacings** option and the weights in the **-spacing_weights** or **-spacing_weight_levels** option must have a one-to-one correspondence.

For the classic router, each entry in the list contains a routing layer name and its spacing in microns. You can specify only a single spacing per layer. The spacing value is a floating point number.

If you do not specify the minimum spacing for a layer, the tool sets the minimum spacing based on the reference rule.

-spacing_weights {*layer_name_and_weight_pairs*}

Sets the weight for the corresponding spacing that is specified in the **-spacings** option.

Each entry in the list contains a routing layer name and its weight list, separated by a space. Each weight value must be a floating point number between 0.0 and 1.0. A value of 1 indicates that the spacing is a hard rule. Smaller values indicate that the corresponding spacing is softer. 0.0 is not a valid value.

The weight values in the **-spacing_weights** argument and the spacing values in the **-spacings** argument must have a one-to-one correspondence.

The **-spacing_weights** and **-spacing_weight_levels** options are mutually exclusive; you can specify only one of these options for a given rule.

This option is supported only by Zroute.

This option is not supported in Design Compiler topographical mode.

-spacing_weight_levels {*layer_name_and_weight_level_pairs*}

Sets the weight level for the corresponding spacing that is specified in the **-spacings** option. This option is an alternative to the **-spacing_weights** option.

Each entry in the list contains a routing layer name and its weight level list, separated by a space. Each weight level value must be a one of low, medium, high, or hard.

- **low**
Treat the spacing rule as a soft rule with low weight.
- **medium**
Treat the spacing rule as a soft rule with medium weight.
- **high**
Treat the spacing rule as a soft rule with high weight.
- **hard**
Treat the spacing rule as a hard rule.

The weight level values in the **-spacing_weight_levels** option and the spacing values in the **-spacings** option must have a one-to-one correspondence.

The **-spacing_weights** and **-spacing_weight_levels** options are mutually exclusive; you can specify only one of these options for a given rule.

This option is supported only by Zroute.

This option is not supported in Design Compiler topographical mode.

-spacing_length_thresholds {*layer_name_and_length_threshold_pairs*}

Sets the length threshold for the corresponding spacing that is specified in the **-spacings** option. Each entry in the list contains a routing layer name and its length threshold list in microns, separated by a space.

Each length threshold value is a floating point number between 0 and 16.3, inclusive. If you specify a length threshold value outside of this range, the command fails.

The length threshold values in the **-spacing_length_thresholds** argument and the spacing values in the **-spacings** argument must have a one-to-one correspondence.

If you do not specify the length threshold values for a layer, the default threshold value is 0.

Zroute reports and tries to fix only those DRC violations where the parallel length of the metal involved in the violation overlaps by at least the specified threshold. Zroute calculates the parallel length using a shape-based formulation. The purpose of this option is to increase the flexibility of DRC convergence but not hurt crosstalk in a significant way.

This option is supported only by Zroute.

This option is not supported in Design Compiler topographical mode.

-shield

Defines shielding with default spacing and default width for the new nondefault routing rule.

This option is not supported in Design Compiler topographical mode.

-shield_widths {*layer_name_and_shield_width_pairs*}

Specifies the shielding width for each named routing layer. Each entry in the list contains a routing layer name and its shielding width in microns, separated by a space. The shielding width portion of the pair is a floating point number. You can specify only a single width per layer.

If you do not specify the shielding width for a layer, the tool sets the shielding width based on the reference rule.

This option is not supported in Design Compiler topographical mode.

-shield_spacings {*layer_name_and_shield_spacing_pairs*}

Specifies the minimum shield spacing allowed between wires for each named routing layer. Each entry in the list contains a routing layer name and its shield spacing in microns, separated by a space. The shield spacing portion of the pair is a floating point number. You can specify only a single spacing per layer.

If you do not specify the shield spacing for a layer, the tool sets the shield spacing based on the reference rule.

This option is not supported in Design Compiler topographical mode.

-snap_to_track

Snaps the shielding wires to the track.

This option is not supported in Design Compiler topographical mode.

-via_cuts {*via_definition_list*}

Creates vias automatically during routing using the via definitions.

For Zroute, each entry in the *via_definition_list* argument consists of the routable via name, its horizontal and vertical site count numbers, and its rotation setting, separated by spaces. The site count portion of the triplet is in the format *mXn* or *mxn*, where *m* is the horizontal site count and *n* is the vertical site count, separated by the X or x character. For a normal via array, all via sites are occupied by cuts, so the total number of cuts is equal to total number of via sites. For a via array with a special pattern, such as a staggered pattern, some via sites are empty, so the total number of cuts is smaller than the total number of via sites. Zroute figures out how many cuts are needed from the specified via array size *mXn* and the required via array pattern. The rotation portion of the triplet is specified by R or NR, where R indicates a rotated via and NR indicates an unrotated via. The default rotation value is NR. Only those modes and rotations that are explicitly specified are allowed by the nondefault routing rule. For example, to allow all via array modes (swapped and unswapped) and rotations (rotated and unrotated) for a 1x2 via array named VIAHH, you must enter the following definition:

```
-via_cuts {{VIAHH 1x2 NR} {VIAHH 1x2 R} {VIAHH 2x1 NR} {VIAHH 2x1 R}}
```

For the classic router, each entry in the *via_definition_list* argument consists of the routable via name and its horizontal and vertical cut count numbers, separated by a space. The cut number is in the format *mXn* or *mxn*, where *m* is the horizontal cut number and *n* is the vertical cut number. The classic router supports only a single via definition for each layer; if you specify multiple vias for a given layer, the first definition is used and additional definitions are ignored. If you specify rotation information, it is ignored by the classic router.

You must ensure that the specified vias exist in the reference routing rule; otherwise, the command fails.

The **-via_cuts** and **-cuts** options are mutually exclusive; you can specify only one.

This option is not supported in Design Compiler topographical mode.

-cuts {*cut_definition_list*}

Creates vias automatically during routing using the general cut definitions.

Each entry in the *cut_definition_list* argument consists of the cut layer name and the cut pair lists, separated by spaces. Each cut pair consists of the cut name and the number of cuts, separated by spaces. The cut name is defined in the Layer section of the technology file along with the width and height information for the cut. The **define_routing_rule** command searches for proper {via, rowXcolumn, rotation} solutions (the description for the **-via_cuts** option for more information) without violating any nondefault rules or technology file rules.

The solutions must satisfy the following conditions:

1. The cut sizes of the vias match the width and height of the specified cut name.
2. The solutions are a subset of the contact codes defined in the fatTblFatContactNumber attribute in the Layer section of the technology file required by the metal width and number of cuts.
3. The cut number is the larger of the number defined in the nondefault routing rule and the number defined in the fatTblFatContactMinCuts attribute in the Layer section of the technology file.

For example, assume the following information is defined in the technology file:

```
Layer "VIA1" {
  fatTblThreshold      = ( 0, 0.181, 0.411 )
  fatTblFatContactNumber = ( "2,3,4,5,6 ", "5,6,20", "5,6,20" )
  fatTblFatContactMinCuts = ( "1,1,1,1,1", "1,1,1", "2,2,2" )

  cutNameTbl          = ( Vsq, Vrect )
  cutWidthTbl         = ( 0.05, 0.05 )
  cutHeightTbl        = ( 0.05, 0.13 )
  ...
}
```

To specify all cuts that match the size requirement of Vrect, meet the fat table rules according to the upper and lower metal widths, and the cut number is at least 1, enter the following definition:

```
define_routing_rule ruleA -widths { M1 0.2 M2 0.25 } -cuts {VIA1 {Vrect 1}}
```

The **define_routing_rule** command automatically populates all cuts that meet the requirements.

The **-via_cuts** and **-cuts** options are mutually exclusive; you can specify only one.

This option is supported only by Zroute.

This option is not supported in Design Compiler topographical mode.

-taper_level *tapering_level*

Defines the tapering level.

Valid values for the tapering level are

- 0 (the default)
Enables pin-based tapering.
- 1
Disables tapering on all pins.
- Positive integer greater than 1 (classic router only)
For the classic router, specifying a positive integer greater than one enables fanout-based tapering with up to the specified number of widths. In this case, the router uses the default width at the input pin and the maximum width, as defined in the

routing rule, is used at the output pin and at the top-level pin.

Zroute does not support this value. To specify the tapering wire width for Zroute, use the **set_route_zrt_detail_options** command to set the **-pin_taper_mode** Zroute detail route option. To specify the tapering distance for Zroute, use the **-taper_distance** option.

This option is not supported in Design Compiler topographical mode.

-taper_distance *tapering_distance*

Sets the maximum length in microns for tapering wires.

To disable tapering, set this option to 0.

If you do not specify this option, the default tapering distance is 10 times the mean number of tracks for all routing layers and the default tapering wire width is the default width for the respective layer.

For Zroute, you can change the tapering wire width to the pin width instead, by using the **set_route_zrt_detail_options** command to set the **pin_taper_mode** Zroute detail route option to **pin_width**.

For the classic router, you can change the tapering wire width by using the **-pin_taper** option.

The tapering wire width is the default width for the respective layer, unless you set the **pin_taper_mode** Zroute detail route option.

This option is not supported in Design Compiler topographical mode.

-driver_taper_distance *driver_tapering_distance*

Sets the maximum length in microns for tapering wires from driver pins.

This option is not supported in Design Compiler topographical mode.

-multiplier_width *layer_width*

Defines the layer width multiplier. You do not have to know the units or exact metal width currently defined in the library.

For example, to define double width, specify **-multiplier_width 2.0**.

-multiplier_spacing *layer_spacing*

Defines the layer spacing multiplier. You do not have to know the units and exact layer spacing currently defined in the library.

For example, to define double spacing, specify **-multiplier_spacing 2.0**.

-double_pattern_mask_constraints *layer_and_constraint_pairs*

Defines the double-patterning mask constraint for each double-patterning routing layer.

Each pair in the list contains a routing layer name and its double-patterning mask constraint. The routing layer name must be one of the double-patterning layers specified in the technology file. The valid double-patterning mask constraints are **mask1_hard**, **mask1_soft**, **mask2_hard**, **mask2_soft**, **same_mask**, and **any_mask**.

The usage of the mask constraint values is as follows:

- Use both **mask1_xxx** and **mask2_xxx** if routes must be mapped to two different masks. Otherwise, choose either **mask1_xxx** or **mask2_xxx**.
- The **mask1_hard** and **mask1_soft** values are treated exactly the same by router. The names hard and soft give you the flexibility to denote different hardnesses of mask constraints, if needed. Likewise for the **mask2_hard** and **mask2_soft** values.
- The **any_mask** value means no double-patterning mask constraint is set.

- The **same_mask** value means that mask mapping is required, but the mask number is yet to be determined.

When you set a double-patterning constraint (except for **any_mask**) on a net, it must meet the double-patterning minimum spacing rule; otherwise, the tool reports a "Double pattern hard/soft mask spacing" violation.

This option is not supported in Design Compiler topographical mode.

-via_spacings via_layer_name_pairs_and_spacing

Specifies the via spacing between two via layers using the following format:

```
{ {layer1 layer2 spacing} ... }
```

The two via layers can be the same layer or different layers.

The minimum spacing between vias on any layer combinations not specified in this option is determined from the Milkyway technology file. For vias on the same layer, the router uses the minimum spacing defined in the Layer section for the via layer. For vias on different layers, the router uses the minimum spacing defined in the DesignRule section for the via layer combination.

This option is supported only by Zroute.

This option is not supported in Design Compiler topographical mode.

-rdl_taper_distance rdl_tapering_distance

Specifies the maximum length in microns for tapering wires in the redistribution layer.

The default is 0 and the wires are not tapered.

This option is supported only by the flip-chip router.

This option is not supported in Design Compiler topographical mode.

-rdl_taper_widths {layer_name_and_rdl_tapering_width_pairs}

Specifies the tapering width for each redistribution layer using the following format:

```
{ {layer width} ... }
```

If you do not specify this option, the flip-chip router uses the size of the via or pin as the wire width.

This option is supported only by the flip-chip router.

This option is not supported in Design Compiler topographical mode.

-taper_over_pin_layers number_of_layers

Enables layer-based pin tapering and specifies the number of metal layers over the pin layer that the router can use for tapering.

For example, if pin is on M1 and you specify **-taper_over_pin_layers 2**, the tool uses the M1 and M2 layers for tapering, but not the M3 layer.

The default is 0. If both the **-taper_over_pin_layers** and **-taper_under_pin_layers** options are set to zero, pin tapering is disabled on the pin layer. If either option is set to a nonzero value, the pin layer is considered a taper layer.

Layer-based pin tapering takes precedence over distance-based pin tapering. If you specify the **-taper_distance** option with this option, the tool ignores the taper distance and issues a ZRT-548 warning message.

This option is not supported in Design Compiler topographical mode.

-taper_under_pin_layers number_of_layers

Enables layer-based pin tapering and specifies the number of metal layers under pin layer that the router can use for tapering.

The default is 0. If both the **-taper_over_pin_layers** and **-taper_under_pin_layers** layers are set to zero, pin tapering is disabled on the pin layer. If either option is set to a nonzero value, the pin layer is considered a taper layer.

Layer-based pin tapering takes precedence over distance-based pin tapering. If you specify the **-taper_distance** option with this option, the tool ignores the taper distance and issues a ZRT-548 warning message.

-single_side_spacing

Specifies to create the rule with single-side spacing, which limits spacing to one side of the net.

DESCRIPTION

This command defines design-specific, nondefault routing rules that are stored in the design database.

Use the **define_routing_rule** command to define width and spacing rules and the via types associated with them, as needed, without predefining them in the physical library.

Use the **set_net_routing_rule** command to assign nondefault routing rules to specific nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines a nondefault routing rule named `new_rule` that uses the default routing rules as the reference rule:

```
prompt> define_routing_rule new_rule -default_reference_rule \
        -widths {m1 0.8 m4 0.9} -spacings {m1 1.0 m4 1.0}
```

The following example defines a nondefault routing rule named `new_rule` that specifies multiple contacts with both rotated and unrotated vias for certain layers. Note that multiple contacts and rotation are supported only for Zroute.

```
prompt> define_routing_rule new_rule \
        -via_cuts {{VIA1_HH 2x1 NR} {VIA1_HH 2x1 R}
                  {VIA1_HV 2x1 NR} {VIA1_HV 2x1 R}}
```

The following example reports the nondefault rule named `new_rule`:

```
prompt> define_routing_rule new_rule
prompt> report_routing_rules new_rule
```

The following example defines a nondefault rule named `new_rule` that specifies spacings, spacing weights, and spacing length thresholds:

```
prompt> define_routing_rule new_rule \
        -spacings {m1 {0.09 0.15 0.2} m2 {0.09 0.15 0.2} m3 {0.09 0.15 0.2} } \
        -spacing_weights {m1 {1 0.5 0.2} m2 {1 0.5 0.2} m3 {1 0.5 0.2}} \
        -spacing_length_thresholds {m1 {0.01 0 0} m2 {0.01 0 0} m3 {0.01 0 0}}
```

The following example defines double-patterning mask constraints for the double-patterning layers.

```
prompt> define_routing_rule color1 \
```

```
-double_pattern_mask_constraints {M01 same_mask M02 mask1_hard \  
M03 mask1_soft M04 mask2_hard M05 mask2_soft}
```

The following example defines via spacing between same via layers and different via layers.

```
prompt> define_routing_rule -via_spacings {{v1 v1 2.3} {v1 v2 3.2}}
```

The following example enables layer-based tapering. When routing a pin with this routing rule, if the pin is on M2, M2 and M3 are the taper layers; there is no tapering on any layer below the pin layer.

```
prompt> define_routing_rule CLK_NDR \  
-taper_over_pin_layers 2 -taper_under_pin_layers 0
```

SEE ALSO

```
remove_routing_rules(2)  
report_net_routing_rules(2)  
report_routing_rules(2)  
set_net_routing_rule(2)
```

define_scaling_lib_group

Defines a scaling library group to support voltage and/or temperature scaling.

SYNTAX

```
status define_scaling_lib_group  
  [-name name]  
  [lib_file_names]
```

Data Types

```
name          string  
lib_file_names list
```

ARGUMENTS

-name *name*

Specifies a name for the scaling library group. The name can be used in a subsequent **set_scaling_lib_group** command.

lib_file_names

Specifies the set of library files that comprise the scaling library group. Specify the names of files that can be found using the **search_path** variable.

DESCRIPTION

The **define_scaling_lib_group** command defines a group of libraries that the tool uses for interpolation to perform voltage and/or temperature scaling.

You can define more than one group to cover different portions of a design, but the groups cannot share libraries. Each library can belong to no more than one scaling library group.

A minimum of two libraries is required for one-dimensional scaling, such as voltage or temperature. A minimum of four libraries is required for two-dimensional scaling, such as voltage and temperature.

The scaling library group defined in the current session is not written back to the design. In a new session, the same scaling library group settings must be defined again.

You should use the **set_scaling_lib_group** command to set the scaling library group defined on the design objects, so that the tool can perform interpolation between libraries in this group for voltage and/or temperature. You should define a scaling library group only if you intend to use it in the current session.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines a scaling library group for two voltage domains about the nominal:

```
prompt> define_scaling_lib_group -name g1 { 0.9.db 1.1.db 1.3.db }
```

SEE ALSO

- `create_scenario(2)`
- `remove_scaling_lib_group(2)`
- `set_operating_conditions(2)`
- `set_scaling_lib_group(2)`

define_test_mode

Defines a test mode to be created during DFT synthesis.

SYNTAX

```
status define_test_mode
    test_mode_label
    [-encoding {port_name 0 | 1, ...}]
    [-usage mode_purpose]
    [-target core_mode_pair_list]
    [-transparent_mode_of parent_mode_name]
```

Data Types

```
test_mode_label    string
mode_purpose        string
core_mode_pair_list list
parent_mode_name  string
```

ARGUMENTS

test_mode_label

Specifies a user-defined text string containing only alphanumeric characters (a-z, A-Z, 0-9) and the underscore (_).

-encoding {port_name 0 | 1, ...}

Specifies a list of ports and the binary values that select a particular test mode. The encoding argument consists of a port name and a binary value pair that specify the test-mode port and the bit values to be used in a particular test mode. The encoding port pairs for a test mode must contain all pairs to be used in the design. See the **EXAMPLES** section for more information.

Before a port can be used for a test-mode port, it must be defined as a test-mode port using the **set_dft_signal** command:

```
set_dft_signal -type TestMode
```

-usage *mode_purpose*

Specifies the mode type for a test mode. The supported mode types are **scan_compression**, **scan**, **wrp_if**, **wrp_of**, **wrp_safe**, and **logicbist**.

- **scan** is the basic scan mode operation. The scan chains are driven directly by top-level scan-in ports and they in turn drive top-level scan-out ports. This mode is used for testing all logic internal to the core. This is the default mode if none is specified.
- **scan_compression** is the DFTMAX compressed scan mode of operation. In this mode, the compressed scan chains are driven by the load compressors and the scan chains drive the unload compressors. This mode is used for testing all logic

internal to the core.

For backward compatibility, you can also use this usage to define DFTMAX Ultra compression modes if the design does not also contain DFTMAX compression modes.

- **streaming_compression** is the DFTMAX Ultra compressed scan mode of operation. In this mode, the compressed scan chains are driven by the streaming load compressors and the scan chains drive the streaming unload compressors. This mode is used for testing all logic internal to the core.
- **wrp_if** is the inward facing, or **INTEST** mode of wrapper operation. This mode is used for testing all logic internal to the core. In this mode, wrappers are enabled and configured to drive and capture data within the design in conjunction with the internal scan chains.
- **wrp_of** is the outward facing, or **EXTEST** mode of wrapper operation. This mode is used for testing all logic external to the design. Wrappers are enabled and configured to drive and capture data outside of the design. In this mode the internal chains are disabled.
- **wrp_safe** is the safe wrapper mode. In this mode the internal chains are disabled and the internal core is protected from toggle activity. This mode is optional and provides isolation of the core while other cores are being tested. When active, safe mode enables driving steady states into or out of the design.
- **logicbist** is the LogicBIST self-test compression mode, which implements a digital logic built-in self-test capability. In this mode, the compressed scan chains are driven by a pseudo-random pattern generator for a fixed number of patterns. A multiple-input signature register (MISR) captures the resulting scan data, then compares it against an expected signature value to generate a PASS/FAIL result.

-target core_mode_pair_list

Specifies test-mode assignments to use for DFT-inserted cores.

This option overrides the default test-mode assignment for DFT-inserted cores. Specify a list of core and test-mode pairs to use for the top-level test mode being defined; each pair consists of a core instance name and a core test-mode name separated by a colon (:). For designs with scan compression, you can also include the name of the current design, without a colon or test-mode name, to specify that the top-level logic should be active and tested.

If a core is targeted in some test modes but not others, it is inactive in the test modes where it is not targeted. This is known as sparse targeting. (To completely exclude a core from all top-level test modes, use the **-exclude_elements** option of the **set_scan_configuration** command.)

Untargeted cores (not included in any target list across all test modes) are tested in top-level modes where the top-level logic is tested.

-transparent_mode_of parent_mode_name

Specifies the parent mode for the transparent test mode being defined.

A transparent mode is derived from an inward-facing parent test mode of a core-wrapped design. It is identical to its parent mode, except that

- Wrapper chains are treated as regular scan chains.
- Dedicated wrapper cells are logically transparent, although their flip-flops remain in the wrapper chains and act as observe test points on I/O paths.
- Any scan compression codecs from the referenced inward-facing mode are reused.
- Feedthrough chains drive the outward-facing wrapper scan I/Os in transparent mode.

A transparent mode allows a core-wrapped design to be tested top-down flat from a higher hierarchy level.

A transparent mode inherits all DFT configuration information from its parent mode. Do not apply any other DFT configuration

commands specifically to a transparent test mode; they are ignored. Do not specify a base mode for a transparent compressed scan mode.

DESCRIPTION

The **define_test_mode** command allows you to define a test mode for DFT synthesis. Inference for existing scan multimode designs is not supported at this time.

The information associated with a *spec* mode is interpreted as a specification of a test mode to be synthesized. This information is used during the DFT architect and insertion process, while running the **preview_dft** or the **insert_dft** command. The specification information for a mode consists of protocols, port attributes, and the configuration information specific to the DFT to be inserted.

Perform multimode specifications in the following order:

1. Define TestMode signals with **set_dft_signal -test_mode all**.
2. Define all test modes, their usage, and their encoding with **define_test_mode**.
3. Define clocks, asynchronous signals, and constants common to all test modes with **set_dft_signal -test_mode all**.
4. Define mode-specific signals with **set_dft_signal -test_mode test_mode_name**.
5. Specify scan paths to be used in all test modes with **set_scan_path -test_mode all**.
6. Specify scan paths to be used in specific test modes with **set_scan_path -test_mode test_mode_name**.

Note that when the **define_test_mode** command is used, it will set the *current_test_mode*. Then if any of the commands listed below are used without the **-test_mode** option, the spec will not be applied to all test modes. Instead, the spec will be applied only to the mode that was last defined with **define_test_mode**. The affected commands are

set_scan_configuration

set_scan_path

set_dft_signal

set_scan_configuration_compression

Be sure to use the **-test_mode** option with all of these commands once the **define_test_mode** command has been issued. If the spec is to be applied to all modes use **-test_mode all**. If the spec is to be applied to a specific test mode, use **-test_mode test_mode_name**.

Use the **remove_test_mode** command to remove a test mode and all associated information.

EXAMPLES

The following example defines three test modes in a DFTMAX compressed scan flow. The modes are ScanCompression_mode, Internal_scan1, and Internal_scan2. The encoding values are also specified.

```
prompt> set_dft_signal -view spec -type TestMode \
        -port [list i_trdy_de i_trdy_dd i_cs]
```

```
prompt> define_test_mode ScanCompression_mode \
  -usage scan_compression \
  -encoding {i_trdy_de 0 i_trdy_dd 0 i_cs 1}

prompt> define_test_mode Internal_scan1 -usage scan \
  -encoding {i_trdy_de 0 i_trdy_dd 1 i_cs 0}

prompt> define_test_mode Internal_scan2 -usage scan \
  -encoding {i_trdy_de 0 i_trdy_dd 1 i_cs 1}
```

The following example defines 9 wrapper modes:

```
prompt> set_dft_signal -view spec -type TestMode \
  -port [list i_trdy_de i_trdy_dd i_cs]

prompt> define_test_mode burn_in -usage scan \
  -encoding {i_trdy_de 0 i_trdy_dd 0 i_cs 0 i_wr 1}

prompt> define_test_mode domain -usage scan \
  encoding {i_trdy_de 0 i_trdy_dd 0 i_cs 1 i_wr 0}

prompt> define_test_mode my_wrp_if -usage wrp_if \
  encoding {i_trdy_de 0 i_trdy_dd 0 i_cs 1 i_wr 1}

prompt> define_test_mode my_wrp_if_delay -usage wrp_if \
  -encoding {i_trdy_de 0 i_trdy_dd 1 i_cs 0 i_wr 0}

prompt> define_test_mode my_wrp_if_scl_delay -usage wrp_if \
  -encoding {i_trdy_de 0 i_trdy_dd 1 i_cs 0 i_wr 1}

prompt> define_test_mode my_wrp_of -usage wrp_of \
  -encoding {i_trdy_de 0 i_trdy_dd 1 i_cs 1 i_wr 0}

prompt> define_test_mode my_wrp_of_delay -usage wrp_of \
  -encoding {i_trdy_de 0 i_trdy_dd 1 i_cs 1 i_wr 1}

prompt> define_test_mode my_wrp_of_scl_delay -usage wrp_of \
  -encoding {i_trdy_de 1 i_trdy_dd 0 i_cs 0 i_wr 0}

prompt> define_test_mode my_wrp_safe -usage wrp_safe \
  -encoding {i_trdy_de 1 i_trdy_dd 0 i_cs 0 i_wr 1}
```

The following example uses the **-target** option to override default name-based test-mode assignment for DFT-inserted cores:

```
prompt> define_test_mode STD -usage scan \
  -target {IPcore:STD MMcore:STD top}

prompt> define_test_mode COMP1 -usage scan_compression \
  -target {IPcore:DFTMAX1 MMcore:CMP top}

prompt> define_test_mode COMP2 -usage scan_compression \
  -target {IPcore:DFTMAX2 MMcore:CMP top}
```

SEE ALSO

current_test_mode(2)
insert_dft(2)
list_test_modes(2)
preview_dft(2)
remove_test_mode(2)
set_dft_signal(2)
set_scan_compression_configuration(2)
set_scan_configuration(2)
set_scan_path(2)
set_streaming_compression_configuration(2)

define_user_attribute

Defines a new user-defined attribute.

SYNTAX

```
string define_user_attribute  
-type string | int | float | double | boolean  
-classes class_list  
[-range_min min]  
[-range_max max]  
[-one_of values]  
[-quiet]  
attr_name  
[-import]
```

Data Types

<i>class_list</i>	list
<i>min</i>	double
<i>max</i>	double
<i>values</i>	list
<i>attr_name</i>	string

ARGUMENTS

-type string | int | float | double | boolean

Specifies the data type of the attribute.

-classes *class_list*

Specifies the classes for the new user-defined attribute. Valid classes are design, port, cell, net, etc.

-range_min *min*

This option is only for PT compatibility.

-range_max *max*

This option is only for PT compatibility.

-one_of *values*

This option is only for PT compatibility.

-quiet

Turns off the warning message that would otherwise be issued if the attribute or classes are improper.

attr_name

Specifies the name of the attribute.

DESCRIPTION

This command defines a new attribute. Use the **list_attributes** command to list the attributes that you have defined.

The definition for a user-defined attribute can be persistent if it has been set on a specified design object and stored in the database.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines an attribute named attr_1, which has a data type of double with a minimum value of 2 and a maximum value of 3.2 and can be set on cells or nets.

```
prompt> define_user_attribute -class {cell net} -type double \  
-range_max 3.2 -range_min 2 attr_1  
attr_1
```

The following example defines an attribute named attr_2, which has a data type of string with valid values of true or false and can be set on nets.

```
prompt> define_user_attribute -class net -type string \  
-one_of {true false} attr_2  
attr_2
```

The following example shows how to list the attribute definitions using the **list_attributes** command:

```
prompt> list_attributes  
*****  
Report : attribute definition  
Design :  
Version: E-2010.12  
Date  : Tue Sep 28 19:23:55 2010  
*****
```

Attributes:
r - read-only
u - user-defined

Attribute name	Class	Type	Attributes

attr_1	cell	float	u
attr_1	net	float	u
attr_2	net	string	u

1

SEE ALSO

get_attribute(2)
list_attributes(2)
remove_attribute(2)
set_attribute(2)

delete_operating_conditions

Deletes a specific set of operating conditions from a library.

SYNTAX

```
status delete_operating_conditions  
-library library_name  
-name op_cond_name
```

Data Types

```
library_name  string  
op_cond_name string
```

ARGUMENTS

-library *library_name*

Specifies the name of the library that stores the operating conditions.

-name *op_cond_name*

Specifies the name of the set of operating conditions.

DESCRIPTION

The **delete_operating_conditions** command deletes a specific set of operating conditions from the specified library.

To create a new set of operating conditions, use the **create_operating_conditions** command.

To set operating conditions on the current design, use the **set_operating_conditions** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example deletes a specific set of operating conditions called *BEST* in the *a_lib* library:

```
prompt> delete_operating_conditions -library a_lib -name BEST
```

SEE ALSO

`create_operating_conditions(2)`
`set_operating_conditions(2)`

derive_constraints

Propagates design environment, constraints, and attribute settings from the top-level design to the specified subdesigns.

SYNTAX

```
status derive_constraints
  [-attributes_only]
  [-verbose]
  [-budget]
  cell_list
```

ARGUMENTS

-attributes_only

Propagates only the attributes. By default, both constraints and attributes are propagated.

-verbose

Enables additional messages during propagation.

-budget

Performs RTL budgeting to come up with delay constraints.

cell_list

Specifies the list of cells to which the environment is to be propagated. The following validations are performed on the specified cells:

- Cell is hierarchical
 - Cell is unique or is a master instance
 - Reference design is in Design Compiler memory
-

DESCRIPTION

This command propagates design environment, constraints, and attribute settings from the top-level design to the specified subdesigns. Although this command works on both mapped and unmapped designs, design budgeting provides more accurate constraints for mapped designs.

This command uses the **target_library** variable. Ensure that this variable is set before running the command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

SEE ALSO

`write_environment(2)`
`MasterInstance(3)`
`target_library(3)`

describe_state_transition

SYNTAX

```
status describe_state_transition
      transition_name
      [-object object]
      [-from from_list]
      [-to to_list]
      [-paired {{from_state to_state}*}]
      [-through through_list]
      [-legal | illegal]
      [-illegal]
```

Data Types

```
transition_name  string
from_list        list
to_list          list
through_list    list
```

ARGUMENTS

transition_name

Specifies the name of the transition that will be defined.

-object *object*

Specifies the name of the object to which states the transition will be defined for, the specified object can be any of the following types:

- Power Domain - Supply Set - Power State Group - Power State Table - Supply Port - Supply Net - Power Switch

-from *from_list*

Consists of an unordered list of power state names active before a state transition. An Empty list expands to all named power states for the specified object.

-to *to_list*

Consists of an unordered list of power state names active after a state_transition. An Empty list expands to all named power states for the specified object.

-paired {{*from_state to_state*}*}

Consists of a list of from-state and to-state name pairs.

-through *through_list*

List of intermediate states. This option is only applicable when *-from* and *-to* options are specified. The effective transition sequence is **-from**, **-through**, **-to**. This is a Synopsys specific extension.

-legal | -illegal

These options specify the legality of the transition being defined as either legal or illegal. The default is **-legal**.

DESCRIPTION

This command defines named state transitions between power states of an object. If the specified object doesn't exist or isn't any of the types indicated above, the command issues an error.

The initial and final states of each transition can be specified either using two separate **-from** and **-to** lists or a **-paired** list. If none of these are specified the command issues an error. Furthermore, the same state can't be an initial and final state of the same transition, and specifying this will cause the command to return an error.

Synopsys has a specific extension consisted of the **-through** option, which is a list of intermediate states to the **-from** and **-to** lists. If this option is specified without the **-from** or **-to** option, the command issues an error.

Design Compiler doesn't preserve this command through hierarchical flow, and a warning will be issued from characterize when this command is seen.

Design Compiler will only parse and preserve this command.

EXAMPLES

Following examples show different usages of **describe_state_transition** command and the outcome.

```
prompt> describe_state_transition TURN_ON -object MID_SS -from [list OFF PWR_OFF] -to [list ON PWR_ON] -legal
1
```

```
prompt> describe_state_transition FAILED1 -object TOP_SS -from [list ON PWR_OFF] -to [list ON PWR_ON]
Error: The same state can't be both an initial and final state in a transition. State ON is in both -from and -to state lists. (UPF-898)
```

```
prompt> describe_state_transition FAILED2 -object mid_true/PST2 -from [list OFF PWR_OFF] -through [list ON PWR_ON]
Error: State list -through can't be defined if -from or -to lists are empty. (UPF-897)
```

```
prompt> describe_state_transition FAILED3 -object mid_true/MID2_SS
Error: State lists -from and -to can't be empty if -paired state list is not defined. (UPF-896)
```

```
prompt> describe_state_transition FAILED4 -object NE -from [list OFF PWR_OFF] -to [list ON PWR_ON] -legal
Error: Object NE wasn't found (UPF-899)
```

SEE ALSO

`add_state_transition(2)`

dft_drc

Checks the current design against test design rules.

SYNTAX

```
status dft_drc  
  [-pre_dft]  
  [-verbose]  
  [-coverage_estimate]  
  [-sample percentage]
```

ARGUMENTS

-pre_dft

Specifies that only pre-DFT rules (D rules) are checked. By default, the state of the design determines the rules that are checked. The state of the design is automatically determined using attributes. For scan-routed designs, post-DFT rules are checked; otherwise pre-DFT rules are checked.

-verbose

Controls the amount of detail when displaying violations. If specified, every violation instance is displayed. By default, only the first instance and the number of instances are displayed.

-coverage_estimate

Generates a test coverage estimate at the end of post-DFT DRC design rule checking.

-sample *percentage*

Specifies a sample percent of faults to be considered when estimating test coverage. This option must be used in conjunction with the **-coverage_estimate** option.

DESCRIPTION

This command checks the current design against the test design rules of the scan test implementation specified by the **set_scan_configuration -style** command. If there are unconnected test modes and functional Autofix clocks, they are constrained as needed.

If design rule violations are found, the appropriate messages are generated.

Perform test design rule checking on a design before performing any other DFT Compiler operations, such as **insert_dft**.

This command requires the existence of a valid test protocol. The test protocol can be generated using the **create_test_protocol** command or read using the **read_test_protocol** command.

The severity of violations falls under one of the following categories:

- Information indicates no action is required.
- Warning indicates that you should analyze the violations. These might violate sequential cells resulting in their exclusion from scan chains. However, they do not prevent you from running certain DFT Compiler commands.
- Fatal violations stop you from proceeding. Certain DFT Compiler commands cannot be run.

If you start with a design with pre-existing scan structures, the scan information for inference must be specified with **-view** as **existing_dft** for the **set_dft_signal** and **set_scan_path** commands and run the **create_test_protocol** command.

The **\fb-coverage** option generates test coverage statistics for the current design. The option does not allow you to save or write out test patterns. The number generated is only an estimate and might be different from the one generated by an ATPG tool.

If you are running the command in the Design Vision environment, you can use the violation browser to analyze violations.

EXAMPLES

The following command performs test design-rule checking for the current design, and illustrates the types of messages that are printed:

```
prompt> current_design des1

prompt> set_scan_configuration -methodology full_scan \
    -style multiplexed_flip_flop

prompt> set_dft_signal -view existing_dft -type Constant \
    -port TM1 -active_state
Accepted dft signal specification.
1

prompt> set_dft_signal -view existing_dft -type Constant \
    -port TM2 -active_state
Accepted dft signal specification.
1

prompt> set_dft_signal -view existing_dft -type MasterClock \
    -port CLK1 -timing [list 45 55]
Accepted dft signal specification.
1

prompt> create_test_protocol -infer_asynch
In all_dft mode...

Information: Starting test protocol creation. (TEST-219)
...reading user specified clock signals...
Information: Identified system clock port CLK1 (45.0,55.0). (TEST-265)
...inferring asynchronous signals...
Information: Inferred active low asynchronous control port as1. (TEST-261)
Information: Inferred active low asynchronous control port as2. (TEST-261)
1
```

```
prompt> dft_drc -ver
```

```
In all_dft mode...
```

```
Loading test protocol
```

```
Pre-DFT DRC enabled
```

```
Information: Starting test design rule checking. (TEST-222)
```

```
...basic checks...
```

```
...basic sequential cell checks...
```

```
...checking for scan equivalents...
```

```
...checking vector rules...
```

```
...checking pre-dft rules...
```

```
-----  
Begin Pre-DFT violations...
```

```
Warning: Clock input CP of DFF ff2 was not controlled. (D1-1)
```

```
Pre-DFT violations completed...
```

```
-----  
DRC Report
```

```
Total violations: 1
```

```
-----  
1 PRE-DFT VIOLATION
```

```
1 Uncontrollable clock input of flip-flop violation (D1)
```

```
Warning: Violations occurred during test design rule checking. (TEST-124)
```

```
-----  
Sequential Cell Report
```

```
1 out of 2 sequential cells have violations
```

```
-----  
SEQUENTIAL CELLS WITH VIOLATIONS
```

```
* 1 cell has test design rule violations
```

```
ff2
```

```
SEQUENTIAL CELLS WITHOUT VIOLATIONS
```

```
* 1 cell is a valid scan cell
```

```
ff1
```

```
Information: Test design rule checking completed. (TEST-123)
```

SEE ALSO

`create_test_protocol(2)`

`current_design(2)`

```
insert_dft(2)
preview_dft(2)
read_test_protocol(2)
set_dft_signal(2)
set_scan_path(2)
target_library(3)
```

disconnect_net

Disconnects a net from pins or ports.

SYNTAX

```
status disconnect_net  
  net  
  object_list | -all
```

Data Types

<i>net</i>	string
<i>object_list</i>	list

ARGUMENTS

net

Specifies the net name or net instance name to disconnect. A net must exist in the current design.

object_list

Specifies the pins and ports disconnected from the net. Only pins and ports existing in the current design are specified. Either *object_list* or **-all** must be specified.

-all

Specifies to break all connections on the net. Either **-all** or *object_list* must be specified.

DESCRIPTION

The **disconnect_net** command breaks the connections between a net or a net instance and its pins or ports. The net, pins, and ports are not removed.

This command accepts only scalar (single bit) nets, and not bused nets.

To connect nets, use the **connect_net** command. To display the pins and ports connected to a net, use the **all_connected** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples disconnect nets using the **disconnect_net** command:

```
prompt> disconnect_net NET0 [get_ports A1]  
Disconnecting net 'NET0' from port 'A1'.
```

```
prompt> disconnect_net NET0 [get_pins U1/A]  
Disconnecting net 'NET0' from pin 'U1/A'.
```

The following example breaks all connections on a net using the **disconnect_net** command:

```
prompt> disconnect_net MY_NET_1 -all  
Disconnecting net 'MY_NET_1' from port 'PORT1'.
```

```
prompt> all_connected [get_nets MY_NET_1]
```

SEE ALSO

- `all_connected(2)`
- `connect_net(2)`
- `create_net(2)`
- `current_design(2)`
- `remove_net(2)`

distance

Describes basic command types for describing mask geometry.

SYNTAX

distance

ARGUMENTS

The **distance** command has no arguments.

DESCRIPTION

This command describes basic types of distance for describing mask geometry.

A *distance* is a basic command syntax unit, such as strings and floating-point numbers.

Distances are returned by some commands and some attributes are distances. Distances can be used as arguments to some commands.

Distances can be grouped into lists to form points. A *point* is a Tcl list of two distances, which are interpreted as the x- and y-values of the coordinate. For example, the origin is {0 0}.

A *rectangle* is a Tcl list of two points, which are interpreted as the lower-left and upper-right corners. For example, the unit square is {0 0} {1 1}.

To access a distance when given a point, you might want to use the **lindex** command.

EXAMPLES

The following examples show how to set several uses of distance:

```
prompt> set spacing 12.25
```

```
prompt> set lower_left_x 2.5
```

```
prompt> set lower_left_y 44.125
```

```
prompt> set corner {22.54 1235.75}
```

```
prompt> set base [list $lower_left_x $lower_left_y]
```

```
prompt> set box [list $base [list 222.14 [expr 200.2 * 7]]]
```

```
prompt> get_attribute $cell1 bbox
```

```
prompt> {241.5 700.1} {246.4 703.1}
```

SEE ALSO

`move_objects(2)`

drive_of

Returns the drive resistance value of the specified library cell pin.

SYNTAX

```
float drive_of  
    library_cell_pin  
    [-rise | -fall]  
    [-piece best | worst | average  
    | [-min]
```

Data Types

library_cell_pin string

ARGUMENTS

library_cell_pin

Specifies the name of the library cell pin whose drive value is to be returned. Use the following format to specify this option:

library_name/lib_cell_name/pin_name

The specified library must already be loaded into the shell.

-rise

Specifies that the command is to return the *library_cell_pin* rise drive resistance.

-fall

Specifies that the command is to return the *library_cell_pin* fall drive resistance.

-piece *best* | *worst* | *average*

Specifies to return the **best**, **worst**, or *average_value* of all of the pieces. The *average_value* corresponds to the *average_value* integer index value. This option can be used only with piecewise-linear libraries.

-min

Gets the wire's drive value.

DESCRIPTION

This command returns the drive or wire_drive resistance of the specified library cell pin. The resistance is primarily used as an argument to the **set_drive** command. The **set_driving_cell** command provides a more convenient and accurate method of describing the drive capability of a port.

If you do not use either the **-rise** or **-fall** option, the command returns the largest of the rise or fall drive resistances.

If the command cannot find the specified *library_cell_pin*, it issues an error message and returns 0.0.

Because drive resistances are associated with timing arcs, it is possible to have more than one distinct rise and fall drive associated with a specified library pin. In this case, the command returns the largest resistance.

If the specified library uses the nonlinear (table-lookup) delay model, the command returns an estimated linear drive value. The **set_driving_cell** command handles nonlinear drives and produces more accurate results than the **set_drive** command for these libraries.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command returns the rise drive of the *my_pin* pin of the *lib_cell* cell from the *tech_lib* library:

```
prompt> drive_of -rise tech_lib/lib_cell/pin
```

The following command searches through all of the pieces of the **rise_pin_resistance** attribute on the *my_pin* pin of the *lib_cell* cell. The command returns the worst value for all of the pieces. This assumes that the *tech_lib* library is modeled as a piecewise-linear library; otherwise, the command returns the normal *rise_resistance*.

```
prompt> drive_of -piece "worst" \  
-rise tech_lib/lib_cell/pin
```

SEE ALSO

set_drive(2)
set_driving_cell(2)

duplicate_logic

Duplicates combinational logic paths, given a startpoint pin, an endpoint pin, and, optionally, a list of intermediate (through) pins.

SYNTAX

```
status duplicate_logic
  -from    start_points
  [-through mid_points]
  -to      end_points
  [-location location]
  [-suffix  suffix]
  [-report_only]
```

Data Types

```
start_points  collection
mid_points    collection
end_points    collection
location      string
suffix        string
```

ARGUMENTS

-from *start_points*

Lists the startpoint pins where the paths to be duplicated start. The startpoint pins are expected to be drivers of their paths. It can be a single pin, a list of pins, or a list of pin lists (when the argument is instantiated multiple times in a single call of the command).

-through *mid_points*

Lists the midpoint pins where the paths are passing through. This argument is optional, which means there is no midpoint checking for the paths.

-to *end_points*

Lists the endpoint pins where the paths to be duplicated end. The endpoint pins are expected to be loads of their paths. It can be a single pin, a list of pins, or a list of pin lists (when the argument is instantiated multiple times in a single call of the command).

-location *location*

Specifies the name of a hierarchical cell, inside of which you want the duplicated cells and nets to be located (in a hierarchical context). If not specified, the duplicated cells and nets are located in the same hierarchical units where their originals belonged.

-suffix *suffix*

Specifies an optional suffix that you can append to the names of the original cells, nets, or ports for the construction of the

duplicated cell, net, or port name. The default suffix string is "_dup". However, the command automatically appends a two-digit index to whatever suffix is used in order to differentiate multiple duplications of the same original cell in different calls of the command.

-report_only

Reports only the duplicate logic path finder results without any duplication being done. There are cases where many paths are found between the startpoint and endpoint. The idea of this optional report is to give you enough feedback to help in the choosing of midpoint alternatives to reduce the scope of the duplication.

DESCRIPTION

Duplicates the combinational paths from startpoint pins, passing through optional midpoint pins, to endpoint pins.

If there are multiple paths found for a single **duplicate_logic** command call, the default behavior is to duplicate each involved cell and net only once.

This command can be invoked whenever there is a Design Compiler netlist present. However, only mapped netlists are officially supported.

Be aware that after duplication, the endpoint pins are disconnected from their original driver net, and therefore in some cases you could end up with unloaded logic path tails after the only load has been disconnected.

If you use the command in Design Compiler topographical mode, and physical information is available, duplicated cells are automatically placed in the same physical locations as their originals.

Restrictions

This command does not duplicate paths when any of the following situations occur:

1. When the intermediate cells are any of the following:

- 1.1. Don't Touch
- 1.2. ILM
- 1.3. Black Box
- 1.4. Macro Cell
- 1.5. Sequential Cell

2. When the path crosses hierarchical boundaries and the child hierarchical cell does not allow hierarchical boundary optimization.

3. When the path crosses power domain boundaries.

EXAMPLES

The following example is typical usage. It specifies duplicated combinational paths between IN1 and \

OUT1 ports:"

```
prompt> duplicate_logic -from IN1 -to OUT1
1
```

The following example uses **-report_only** to test how many and how long the paths are between Port A and Port C:

```
prompt> duplicate_logic -from A -to C -report_only  
Valid Path (1 of 2) (4 Nodes): { A/**inside** U2/A U2/Z C/**inside** }  
Valid Path (2 of 2) (6 Nodes): { A/**inside** U3/B U3/Z U2/B U2/Z C/**inside** }  
1
```

The following example uses the **-through** option to reduce the paths to be duplicated:

```
prompt> duplicate_logic -from A -to C -through U2/A -report_only  
Valid Path (1 of 2) (4 Nodes): { A/**inside** U2/A U2/Z C/**inside** }  
1
```

SEE ALSO

echo

Echos arguments to standard output.

SYNTAX

```
string echo  
[-n]  
[arguments]
```

Data Types

arguments string

ARGUMENTS

-n

Suppresses the new line. By default, **echo** adds a new line.

arguments

Specifies the arguments to be printed.

DESCRIPTION

The **echo** command prints out the value of the given arguments. Each of the arguments are separated by a space. The line is normally terminated with a new line, but if the **-n** option is specified, multiple **echo** command outputs are printed onto the same output line.

The **echo** command is used to print out the value of variables and expressions in addition to text strings. The output from **echo** can be redirected using the **>** and **>>** operators.

EXAMPLES

The following are examples of using the **echo** command:

```
prompt> echo  
"Running version" $sh_product_version
```

Running version v3.0a

```
prompt> echo -n "Printing to" [expr (3 - 2)] > foo  
prompt> echo " line." >> foo  
prompt> sh cat foo  
Printing to 1 line.
```

SEE ALSO

sh(2)

elaborate

Builds a design from the intermediate format of a Verilog module, a VHDL entity and architecture, or a VHDL configuration.

SYNTAX

```
status elaborate  
  design_name  
  [-library library_name | -work library_name]  
  [-architecture arch_name]  
  [-parameters param_list]  
  [-file_parameters file_list]  
  [-update]  
  [-ref]
```

Data Types

```
design_name    string  
library_name  string  
arch_name     string  
param_list    list  
file_list     list
```

ARGUMENTS

design_name

Specifies the name of the design to build. The design can be a Verilog module, a VHDL entity, or a VHDL configuration.

-library *library_name*

Specifies the library name to which **work** is to be mapped. By default, **elaborate** looks in the **work** library for the design to be built. Specifying **-library** allows you to temporarily change the library to which **work** is mapped.

-work *library_name*

Specifies an alias used for **-library**.

-architecture *arch_name*

Specifies the name of the architecture. In VHDL, the default architecture is the most recently analyzed architecture.

-parameters *param_list*

Specifies a list of design parameters enclosed in quotes. Parameters within the list must be separated by commas. A specification can be based on parameter order (for example, "8,7,5") or on parameter names (for example, "N=>8,M=>6"). It is acceptable to mix ordered and named parameter specifications, as long as the ordered parameters are listed first. The full *param_list* syntax is

summarized in the "DESCRIPTION" section below.

-file_parameters *file_list*

Specifies a list of files that contain parameter specifications. The specifications are appended to the parameters listed in the **-parameters** option (if present). The syntax of the parameter file is identical to the syntax between the parameter-delimiters (#) in the **-parameters** option, except that the backslashes (\) at the end of lines and the hashes (#) must be omitted. The specified files are searched for in the `search_path`.

-update

Reanalyzes out-of-date intermediate files if the source can be found.

-ref

Elaborates several reference designs that contain bottom design instantiations in the elaborated module, using a hierarchical elaboration flow. When the top design is instantiated, it includes all the linkage information, including interfaces, modports, and parameters, and the logic for the lower-level designs.

DESCRIPTION

The **elaborate** command builds a design from its intermediate representation using the specified parameters.

The syntax of the parameter specifications includes a subset of VHDL syntax plus the ``h` constant format of Verilog. VHDL accepts all but the ``h` format. Verilog accepts strings, integers, and ``h` values. Define each parameter only once; an error message is generated if a parameter is defined more than once.

Parameters are not case-sensitive, so `NUM` is the same as `num`.

For the **-parameters** option (but not in files specified by **-file_parameters**), the parameter specification must be specially delimited. The parameter specification can be enclosed in double quotation marks ("), but this is not advised if the specification itself includes string values. Instead, use the special parameter-delimiter, which brackets the parameter specification syntax. The parameter-delimiter starts and ends with ``#` (hash). For multi-line strings put backslashes (\) as the last character in the line. This is because `"#"` is properly recognized as a string constant.

Spaces can be inserted in the middle of the parameter string to make your script easier to read, but you must either enclose the parameter string in double quotation marks ("), or escape the spaces using backslashes (\).

In the following example, the same parameter is passed three times, one without intermediate spaces and two using the appropriate syntax. All three cases are equivalent.

```
prompt> elaborate TOP -param width=>32,ports=>8
```

```
prompt> elaborate TOP -param "width => 32, ports => 8"
```

```
prompt> elaborate TOP -param #width\ =>\ 32,\ ports\ =>\ 8#
```

In the following example, the second # in the first line does not terminate the parameter:

```
prompt> elaborate MY_DES -param #words=>4,str => "#",\
name=>"my_name",\
matrix=>"1010101010001"&\
"0001010101010"#
```

The formal syntax for the parameter specification is as follows:

```
parameter_list: parameter_spec [ , parameter_spec ]*
```

```
parameter_spec: parameter_name => value_spec
                parameter_name = value_spec
                value_spec
```

```
value_spec:    " string_value "
               ' character_value '
               integer_value
               enum_value
               ( value_spec [ , value_spec ]* )
               integer_value 'h' hex_value
```

When the **-ref** option is specified, the **elaborate** command scans the design and groups the bottom design instantiations into several reference designs. Each reference design contains all of the instantiations of a bottom design type.

For example, if a *top* design contains several instances of bottom designs, name *bot1* and *bot2*, the **-ref** option generates two reference designs named *top_bot1* and *top_bot2*, that contain all of the instantiations of each type of lower design, respectively. The instance parameters and SV interface specializations for SystemVerilog designs are preserved in the reference design information.

Later, the reference design can be loaded and linked, forcing the elaboration of all of the specialized information required from the *top* design. The already-elaborated bottom designs can be used in a hierarchical elaboration flow.

During elaboration, the **link** command is invoked to resolve all instances in the designs. When instances cannot be resolved, the **elaborate** command issues error messages for those instances, returning a 1 status, and continues to resolve all other instances without aborting the elaboration process. Link error messages issued during elaboration are considered as warning messages, and an explicit **link** command should be executed after reading the design.

For more information about the hierarchical elaboration flow for SystemVerilog designs with interfaces, see the *SystemVerilog User Guide*.

EXAMPLES

The following example builds the design *mult* with *N* parameter set to 8, and *M* set to 3. The **-update** switch causes all out-of-date HDL files to be automatically reanalyzed.

```
prompt> elaborate -update mult -parameters "N=8,M=3"
Information: Re-analyzing the out of date package 'USER_TYPES'. (LBR-15)
Information: Re-analyzing the out of date entity 'MULT'. (LBR-15)
Information: Re-analyzing the out of date architecture 'STRUCTURE(MULT)'. (LBR-15)
Current design is now 'MULT_N8_M3'.
1
```

The following example generates a reference design for hierarchical elaboration using the **-ref** switch. The resulting design is saved to a .ddc file.

```
prompt> elaborate -ref
Running PRESTO HDLC
Presto compilation completed successfully.
Elaborated 1 reference design. Use 'link' to elaborate the required down design.
Current design is now 'top_bottom'.
Ending elaborate -ref mode. Please link the generated designs later
1
```

```
prompt> write -f ddc -hier -o top_bottom.ddc top_bottom
Writing ddc file 'top_bottom.ddc'.
1
```

The following example uses a reference design previously generated with the **-ref** switch, links it to elaborate the bottom designs, includes specialized information required by the top module, and writes out a .ddc file:

```
prompt> analyze -f sverilog bottom.sv
Running PRESTO HDLC
Searching for ./bottom.sv
Compiling source file ./bottom.sv
Presto compilation completed successfully.
1
```

```
prompt> read_ddc top_bottom.ddc
Reading ddc file 'top_bottom.ddc'.
Loaded 1 design.
Current design is 'top_bottom'.
top_bottom
```

```
prompt> link
```

```
Linking design 'top_bottom'
Using the following designs and libraries:
```

```
-----
top_bottom      ./top_bottom.ddc
```

```
Information: Building the design 'bottom' instantiated from design 'top_bottom'
              with the parameters "N=32,M=16". (HDL-193)
Presto compilation completed successfully.
Information: Building the design 'bottom' instantiated from design 'top_bottom'
              with the parameters "M=64". (HDL-193)
Presto compilation completed successfully.
1
```

```
prompt> list_designs
bottom_M64    bottom_N32_M16 top_bottom (*)
1
```

```
prompt> write -f ddc -hier -o bottom.ddc { bottom_M64 bottom_N32_M16 }
Writing ddc file 'bottom.ddc'.
1
```

SEE ALSO

```
analyze(2)
define_design_lib(2)
link(2)
list_designs(2)
read_file(2)
report_design_lib(2)
write_file(2)
```

```
hdlin_auto_save_templates(3)
template_naming_style(3)
template_parameter_style(3)
template_separator_style(3)
```

enable_redirect_bg_commands

Specify the allowed command type for redirect background executable.

SYNTAX

```
string enable_redirect_bg_commands  
[-all] [-read_only]
```

ARGUMENTS

-all

This is the default value. It allows all supported command for redirect background executable.

-read_only

Only allows read_only commands for redirect background executable.

DESCRIPTION

This command will specify the allowed command type for redirect background executable. If the command is not called or called with no option, all supported commands will be allowed for redirect background executable. If *-read_only* option is specified, only the read_only commands are allowed.

SEE ALSO

redirect(2)

enable_write_lib_mode

Enables usage of the **write_lib** command in the tool session.

SYNTAX

status **enable_write_lib_mode**

ARGUMENTS

None

DESCRIPTION

This command enables usage of the **write_lib** command in the tool. By default, the **write_lib** command is disabled to optimize for performance. To enable the **write_lib** functionality, you must run the **enable_write_lib_mode** command at the beginning of the tool session, before you load any .db library files. After it is enabled, using the **write_lib** command writes out the complete library file. You should not perform optimization on the design during the session and you need to exit the tool completely after the library operation is done.

EXAMPLES

The following command enables usage of the **write_lib** command:

```
prompt> enable_write_lib_mode
```

This icc_shell session is for library operations only. You should not perform optimization on the design. Please exit when you are finished with the library operation.

```
1
```

```
prompt> report__write_lib_mode
```

Information: write_lib mode is enabled.

```
1
```

```
prompt> write_lib my_lib
```

```
1
```

SEE ALSO

report_write_lib_mode(2)
write_lib(2)

encrypt_lib

Encrypts a VHDL source library file.

SYNTAX

```
status encrypt_lib  
  file_name  
  [-output encrypted_file]  
  [-format {vhdl | tf | mwlib}]  
  [-key string]
```

Data Types

<i>file_name</i>	string
<i>encrypted_file</i>	string

ARGUMENTS

file_name

Specifies the name of the VHDL file to encrypt.

-output *encrypted_file*

Specifies the output filename to which the encrypted output is to be written. By default, the output file is named *file_name.E* and is written to the current directory.

NOTE: If you specify an output filename that already exists, the command overwrites the file without warning.

DESCRIPTION

This command encrypts a VHDL source library file to protect ASIC vendors' VHDL models. The encrypted file is read and simulated by **vhdlan** and **vhdlsim**, respectively, in the same way as a nonencrypted file.

EXAMPLES

The following example encrypts the technology library *my_file.vhd*. The output file is named *my_file.E* and is written to the current directory.


```
prompt> encrypt_lib my_file.vhd
```

SEE ALSO

`read_lib(2)`

`write_lib(2)`

error_info

Prints extended information on errors from the last command.

SYNTAX

string **error_info**

ARGUMENTS

This **error_info** command has no arguments.

DESCRIPTION

The **error_info** command is used to display information after an error has occurred. Tcl collects information showing the call stack of commands and procedures. When an error occurs, the **error_info** command can help you to focus on the exact line in a block that caused the error.

EXAMPLES

This example shows how **error_info** can be used to trace an error. The error is that the iterator variable "s" is not dereferenced in the 'if' statement. It should be '\$s == "a" '.

```
prompt> foreach s $my_list {
?   if { s == "a" } {
?     echo "Found 'a'!"
?   }
?   }
Error: syntax error in expression " s == "a" "
    Use error_info for more info. (CMD-013)
prompt> error_info
Extended error info:
syntax error in expression " s == "a" "
while executing
"if { s == a } {
  echo "Found 'a'"
}"
```

```
("foreach" body line 2)
invoked from within
"foreach s [list a b c] {
  if { s == a } {
    echo "Found 'a'"
  }
}"
-- End Extended Error Info
```

estimate_fp_black_boxes

Sets the size of a black box based on an estimation of the objects that it will contain when replaced with real logic.

SYNTAX

```
status estimate_fp_black_boxes
  [-sm_size {width height}
  | -polygon {{x1 y1} {x2 y2} ...}
  | -sm_gate_equiv gate_count]
  [-sm_util util]
  [-hard_macros names]
  [-fixed_shape]
  [-reset_shape]
  black_boxes
```

Data Types

<i>width</i>	float
<i>height</i>	float
<i>x1</i>	float
<i>y1</i>	float
<i>x2</i>	float
<i>y2</i>	float
<i>gate_count</i>	integer
<i>util</i>	float
<i>names</i>	string
<i>black_boxes</i>	collection

ARGUMENTS

-sm_size {*width height*}

Specifies the width and height of the black box. The estimated area is determined by multiplying width by height, given by a list of two floating point numbers. If you also use the **-sm_util** option, an equivalent gate count value is calculated internally based on the *util* value. Otherwise, the tool uses a value of 1.0 for utilization when the **-sm_size** option is used, but the **-sm_util** option is not used. The **-sm_size** and **-polygon** options are mutually exclusive: you can use only one. By default, this option is off.

-polygon {{*x1 y1*} {*x2 y2*} ...}

Creates a rectilinear black box based on the specified coordinates. The polygon is defined by specifying the coordinates of all its points. The format is {{*x1 y1*} {*x2 y2*} ... {*x1,y1*}}. Specify the points as absolute coordinates in microns. The **-sm_size** and **-polygon** options are mutually exclusive; you can use only one. When you use the **-polygon** option to estimate black box size, the **-sm_util** setting defaults to 1.0. By default, this option is off.

-sm_gate_equiv *gate_count*

Estimates the area as a soft macro with the specified number of equivalent gates. The command calculates an area by multiplying the specified `gate_count` with the area of the currently stored base gate area (either a default 2-input-nand gate area or one which has previously been updated using the **set_fp_base_gate** command) and then dividing by the currently-stored utilization factor (either the default value of 0.7 or the also-specified `sm_util` value).

Use the **set_fp_base_gate** command to change the area of the equivalent gate.

-sm_util util

Estimates the area as a soft macro with the specified utilization. You must also specify the **-sm_gate_equiv** option with a positive value for `gate_count` when you specify the **-sm_util** option. The default is 0.7 if the **-sm_size** or **-polygon** options are not specified.

-hard_macros names

Specifies a list of reference library hard macro names in a string with the names separated by spaces or commas. Each name can have an optional instance count. The instance count is specified as a number in parentheses () appended to the name. For example, to specify 10 instances of the mem macro use the **estimate_fp_black_boxes -hard_macros {mem(10)}** command.

The size estimation includes the area of all the hard macros in the specified list. This estimation mode does not consider the shapes of the actual hard macros.

-fixed_shape

Marks the black box as fixed shape after estimation.

-reset_shape

Resets the shape to rectangular if it is currently rectilinear.

black_boxes

Specifies the names of the black boxes to estimate.

DESCRIPTION

This command sets the estimated size of a black box based on the objects that it will contain when replaced with real logic. The estimation can be as a soft or hard macro. As a soft macro it has one, but not both, of the following characteristics:

- A size specified by using the **-sm_size** or **-polygon** option
- The number of gate-equivalent cells with utilization specified by using the **-sm_gate_equiv** option.

You can specify the utilization in any of the 3 cases: **-sm_size**, **-polygon**, or **-sm_gate_equiv**. As a hard macro the block has either a specified size set by using the **-sm_size** option, or a number of gate equivalent cells set by using the **-sm_gate_equiv** option.

If you do not use the **-hard_macros** option, then you must use the **-sm_size**, **-polygon**, or **-sm_gate_equiv** option. If you specify both the **-sm_size** and the **-sm_gate_equiv** options together, the `size` value takes precedence and the `gate_count` value is ignored; the `gate_count` value becomes a dependent variable that is calculated based on the `size` value. Similarly, if you specify both the **-polygon** and **-sm_gate_equiv** options together, the `polygon_area` value takes precedence and the `gate_count` value is ignored; the `gate_count` value becomes a dependent variable that is calculated based on the `polygon_area` value.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the 2-input NAND gate equivalent cell to nand2d1 and creates a black box estimate for macro alu1. The macro contains 12000 nand2d1 equivalents with a utilization of 0.7.

```
prompt> set_fp_base_gate -cell {nand2d1}
prompt> estimate_fp_black_boxes -sm_gate_equiv 12000 -sm_util 0.75 \
[get_cells -filter "is_physical_black_box==true" alu1]
```

The following example estimates a black box named alu1 to a rectilinear soft macro.

```
prompt> estimate_fp_black_boxes \
-polygon {{1723.645 1925.365} {1723.645 1722.415} \
{1803.595 1722.415} {1803.595 1530.535} \
{799.915 1530.535} {799.915 1925.365} \
{1723.645 1925.365}} \
[get_cells -filter "is_physical_black_box==true" \
I_ORCA_TOP/I_PCI_TOP] \
[get_cells -filter "is_physical_black_box==true" alu1]
```

SEE ALSO

get_cells(2)
set_fp_base_gate(2)

exit

Terminates the application.

SYNTAX

```
integer exit  
[exit_code]
```

Data Types

exit_code integer

ARGUMENTS

exit_code

Specifies the return code to the operating system. The default value is 0.

DESCRIPTION

This command exits from the application. You have the option to specify a code to return to the operating system.

EXAMPLES

The following example exits the current session and returns the code 5 to the operating system. At a UNIX operating system prompt, verify (**echo**) the return code as shown.

```
prompt> exit 5  
  
% echo $status  
5
```

SEE ALSO

quit(2)

extend_mw_layers

Extends Milkyway database layer number support to 4095 layers, using layers 4001 through 4095 as the system-reserved layers.

SYNTAX

status **extend_mw_layers**

DESCRIPTION

By default, the Milkyway database supports the usage of up to 255 layers, numbered 1 through 255. Layers 1 through 187 are available as user-defined layers and routing layers, while layers 188 through 255 are reserved for the Milkyway system layers.

Milkyway system layers are the layers used to define blockages, route guides, and other physical features that guide the router. The Milkyway system layer numbers are fixed and cannot be assigned or changed by the user. User-defined layers are layers defined in the technology (.tf) file, such as metal layers and text layers.

Some advanced process and routing technologies require more than 255 layers. In these cases, you can extend the number of layers supported in a Milkyway library by executing the **extend_mw_layers** command. In that case, layers 1 through 4000 are available as user-defined layers, while layers 4001 through 4095 are reserved for the Milkyway system layers.

In both the default and extended layer modes, routing layers are restricted to layer numbers 1 through 187. Routing layers are the standard layers used by the IC Compiler router to make interconnections, including wire routes, vias, and contacts.

To create a new Milkyway library using extended layers, you must execute the **extend_mw_layers** command before the **create_mw_lib** command. Once created in extended layer mode, the Milkyway library cannot be changed back to use the default layer mode.

In future sessions, the tool automatically recognizes the layer mode of the Milkyway library and uses the correct Milkyway system layer numbers, from 188 to 255 in the default mode or from 4001 to 4095 in the extended layer mode.

EXAMPLES

The following example shows the usage of the command to extend layer number support to 4095 layers and to use layers 4001 through 4095 as the system-reserved layers in a new Milkyway library.

```
prompt> extend_mw_layers
1
prompt> create_mw_lib -technology mytech.tf my_lib_2
Start to load technology file mytech.tf
...
```

SEE ALSO

`create_mw_lib(2)`

extract_physical_constraints

Extracts physical constraint information from one or more Design Exchange Format (DEF) files.

This command is supported only in Design Compiler in topographical mode.

SYNTAX

```
status extract_physical_constraints  
  def_files  
  [-verbose]  
  [-no_incremental]  
  [-exact]  
  [-allow_physical_cells]  
  [-standard_cell topo | spg]
```

Data Types

def_files list

ARGUMENTS

def_files

Specifies one or more DEF files from which the physical constraint information is extracted. This argument is required.

-verbose

Provides more detailed output during the process of extracting the physical constraints.

-no_incremental

Indicates that the command runs in non-incremental mode. By default, the command runs in incremental mode.

In incremental mode, the placement area is extracted based on the current core area and the site rows in the DEF file.

In non-incremental mode, the placement area is extracted based on the site rows in the DEF file. If there are no site rows in the DEF file, other constraints are not extracted, and the following physical constraints are reset:

- Core area
- Route guide
- Voltage area
- Layer preferred routing direction
- Die area

- Placement area
- Macro location and orientation
- Hard, soft, and partial placement blockages
- Wiring keepouts
- Placement bounds
- Port location
- Preroutes
- Site array information
- Vias
- Routing tracks
- Keepout margins

-exact

Specifies that an exact match is performed between the netlist and DEF file for cell and port names.

By default, a rule-based match is performed, with the underlying match rules determined by the **set_query_rules** command.

-allow_physical_cells

Specifies that physical-only cells are extracted from the DEF file and added to the design. A cell instance is considered a physical-only cell if the **cell_type** attribute of its library cell is **std_filler** or if it has only power and ground pins or no pins at all. Physical-only cells are not written to the ASCII netlist, and they are not passed to IC Compiler through the .ddc file. You can identify physical-only cells in the design by the **is_physical_only** attribute.

In addition to extracting physical-only cells from the DEF file, when you use the **-allow_physical_cells** option, the **extract_physical_constraints** command extracts any cell in the DEF file that is not found in the design if the cell has a fixed location. All other cells in the DEF file that are not considered to be physical-only cells are added to the design as logic cells. These cells can be ROM or RAM cells. When you run the **extract_physical_constraints -allow_physical_cells** command, the tool updates the current design with these new logic cells and includes them in your Verilog netlist.

The tool sets a **logical_cell_from_def** attribute on logic cells that are extracted from a DEF file. You can identify these logic cells by using the following command:

```
prompt> get_cells -hierarchical -filter " logical_cell_from_def == true"
```

By default, the tool ignores physical-only cells in the DEF file.

-standard_cell topo | spg

Reads the standard cell locations specified in the DEF file when you use the ASCII flow.

Specify **topo** if the DEF file was generated by using the Design Compiler topographical flow. Specify **spg** if the DEF file was generated by using the Design Compiler physical guidance flow.

DESCRIPTION

The **extract_physical_constraints** command extracts the physical constraint information from one or more DEF files and applies

these constraints to the design.

The applied constraints are saved in the .ddc format and do not need to be reapplied in a new topographical mode session when the .ddc file is read in.

The saved constraints are recognized only by Design Compiler topographical mode; they have no effect in other tools, such as Design Compiler in wire load mode or IC Compiler.

Note that not all information in DEF files is extracted and applied to the design. Only the following types of information are extracted and applied:

- **Placement area**
Placement area is computed based on the site array definition and is applied to the design.
- **Port location**
Each port with a location and port shape in the DEF file has the location and port shape set on the corresponding port in the design. Ports that do not exist in the design or do not have locations in the DEF file have no effect on the design.
- **Cell location**
Each cell with a location and a FIXED attribute in the DEF file has the location set on the corresponding cell in the design. Cells that do not exist in the design or do not have a location or FIXED attribute in the DEF file have no effect on the design.
- **Placement blockage**
Placement blockages are extracted as placement keepouts and are set on the design. For a single rectangle placement blockage in the DEF file, one placement keepout is created on the design. For placement blockages with multiple rectangles, one placement keepout is created for each rectangle. All created placement keepouts can be hard, soft, or partial keepouts.
- **Site rows**
The site row commands are extracted and applied if they exist in the DEF file.
- **Bounds**
If regions are defined in the DEF file, the placement bounds are extracted. Also, if there are cells in the related group attached to the region, the cells are matched using query rules with the ones in the design and the matched cells are attached to the bounds in the following ways:
 - If there are regions in the design with the same name in the DEF file, in incremental mode, the cells in the related group are attached to the region.
 - Otherwise, the region is created with the same name as in the DEF file and the matched cells in the related group are also attached.
- **Wiring keepouts**
Wiring keepouts are extracted in a manner similar to the way in which placement blockages are extracted.
- **Preroutes**
Preroutes are extracted from DEF and applied by default. Preroutes include net shapes, vias, and via arrays of matched contact codes, as well as via cells.

To check which physical constraints were applied to the design, use the **report_physical_constraints** command. To generate a Tcl script of the physical constraints, use the **write_physical_constraints** command.

EXAMPLES

The following example shows how to extract the physical constraints from a DEF file named in.def:

```
prompt> extract_physical_constraints in.def
```

SEE ALSO

`create_bounds(2)`
`create_net_shape(2)`
`create_placement_blockage(2)`
`create_site_row(2)`
`create_wiring_keepouts(2)`
`report_physical_constraints(2)`
`set_query_rules(2)`
`write_physical_constraints(2)`
`update_bounds(2)`

extract_rc

Executes virtual routing for nets in a design.

SYNTAX

```
status extract_rc
      -estimate
```

ARGUMENTS

-estimate

Performs RC estimation for nets in the design. For RC estimation, the command performs virtual routing and delay calculation and back-annotates the delay and capacitance on the current design. After running this command, you can use the **report_timing** command to see the estimated timing results. In Design Compile in Topographical mode, the **-estimate** option is required for the command to run.

DESCRIPTION

This command performs RC extraction for virtual routing of design nets. It calculates delay based on the Elmore delay model.

If you use the **set_tlu_plus_files** command to specify the TLUPlus technology files, the tool performs extraction based on TLUPlus technology.

If your physical library does not contain extraction parameters, the tool derives the resistance and capacitance values of routes from the available RC estimation method in your physical library. The tool issues the **RCEX-015** information message, which determines the RC estimation method that is used.

The command is available only in Design Compiler in topographical mode.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following example performs virtual routing and updates back-annotated net delays.

```
prompt> extract_rc -estimate
```

SEE ALSO

- read_parasitics(2)
- report_timing(2)
- set_extraction_options(2)
- set_operating_conditions(2)
- set_tlu_plus_files(2)
- write_parasitics(2)

filter

Returns a list of design objects that satisfy a conditional attribute expression.

SYNTAX

```
list filter  
  object_list  
  expression  
  [-regexp]  
  [-nocase]
```

Data Types

<i>object_list</i>	list
<i>expression</i>	string

ARGUMENTS

object_list

Specifies a list of design or library objects to filter.

expression

Specifies a conditional expression used to filter the object list in which attribute names are preceded by @ (at sign). For examples of typical expressions, see the "EXAMPLES" section. For a complete description of conditional expressions, refer to the dc_shell man page.

-regexp

Indicates that the real regular expressions are to be used by the =~ and !~ filter operators. By default, the =~ and !~ filter operators use simple wildcard pattern matching with the '*' and '?' wildcards.

-nocase

When used with the **-regexp** option, the pattern matching is case-insensitive. This option must be used with the **-regexp** option.

DESCRIPTION

The **filter** command selectively searches a group of objects and returns only those objects for which the conditional expression is true. The return of an empty list indicates that there are no objects in *object_list* for which the conditional expression is true. To use **filter** with Boolean values, the correct syntax is **true** and **false**. For a list of Synopsys attributes that can be used with **filter**, refer to

the Design Compiler documentation.

The basic form of the conditional expression is a series of relations joined together with && and || operators. Parentheses are also supported. The basic relation contrasts an attribute name with a value through a relational operator. For example:

```
is_hierarchical == true && area <= 6
```

The relational operators are as follows:

```
== Equal
!= Not equal
> Greater than
< Less than
>= Greater than or equal to
<= Less than or equal to
=~ Matches pattern
!~ Does not match pattern
```

The basic relational rules are as follows:

- String attributes can be compared with any operator.
- Numeric attributes cannot be compared with pattern match operators.
- Boolean attributes can only be compared with == and !=. The value can be only true or false.

Existence relations determine if an attribute is defined or not defined for the object. For example:

```
sense == setup_clk_rise && defined(@sdf_cond)
```

The existence operators are as follows:

```
defined
undefined
```

These operators apply to any attribute as long as it is valid for the object class.

For a complete description of conditional expressions, see the documentation for the tool.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care when you quote the filter *expression*. Using rigid quoting around regular expression pattern is recommended. See the "EXAMPLES" section for more details. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding "." to the beginning or end of the expressions as needed. You can make the regular expression search case-insensitive by using the **-nocase** option.

EXAMPLES

The following example uses the **filter** command to return the bidirectional ports of the current design:

```
prompt> filter [get_ports *] \
    "@port_direction == inout"
{B1 B2}
```

The following example finds all of the designs in the system that are programmable logic arrays or state machines:

```
prompt> filter [get_designs *] \
    "@design_type == pla || @design_type == fsm"
```

```
{CONTROL COUNTER}
```

The following example finds all nets for which **dont_touch** is set to **true**:

```
prompt> filter [get_nets *] \
        "@dont_touch == true"
{N1 N9}
```

The following example finds all ports from in[18] to in[23]. The placement of double quotes and backslashes is necessary to avoid syntax errors.

```
prompt> filter [get_ports] \
        -regexp {@name=~ "in\[18-23\]"}
{in[23] in[22] in[21] in[20] in[19] in[18]}
```

The following example finds the *U2* and *U4* cells, using the **-nocase** option:

```
prompt> filter [get_cells] -regexp \
        -nocase {@name =~ u[2,4]}
{U2 U4}
```

The following example uses wildcard characters to match the name. Wildcard characters can be used only with the **=~** and **!~** operators.

```
prompt> filter [get_cells] {@name =~ U?}
{U1 U2 U3 U4}
```

The following example finds all cells in the system that have the ****logic_0**** reference name. The ***** and **?** characters have special meanings when used with the **=~** and **!~** operators, allowing you to use the **==** operator for an exact match.

```
prompt> filter [get_cells] {@ref_name == **logic_0**}
```

SEE ALSO

```
dc_shell(1)
current_design(2)
filter_collection(2)
get_attribute(2)
remove_attribute(2)
set_attribute(2)
attributes(3)
verbose_messages(3)
```

filter_collection

Filters an existing collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

```
collection filter_collection
  [-regex]
  [-nocase]
  collection1
  expression
```

Data Types

<i>collection1</i>	collection
<i>expression</i>	string

ARGUMENTS

-regex

Specifies that the =~ and !~ filter operators will use real regular expressions. By default, the =~ and !~ filter operators use simple wildcard pattern matching with the * and ? wildcards.

-nocase

Makes the pattern match case-insensitive.

collection1

Specifies the base collection to be filtered. This collection is copied to the result collection. Objects are removed from the result collection if they are evaluated as **false** by the conditional *expression* value. Substitute the collection you want for *collection1*.

expression

Specifies an expression with which to filter *collection1*. Substitute the string you want for *expression*.

DESCRIPTION

Filters an existing collection, resulting in a new collection. The base collection remains unchanged. In many cases, application commands that create collections support a *-filter* option that filters as part of the collection process, rather than after the collection has been made.

This type of filtering is almost always more efficient than using the **filter_collection** command after a collection has been formed. The **filter_collection** command is most useful if you plan to filter the same large collection many times using different criteria.

The **filter_collection** command results in either a new collection or an empty string. A resulting new collection contains the subset of the objects in the input *collection1*. A resulting empty string (the empty collection) indicates that the *expression* filtered out all elements of the input *collection1*.

The basic form of the conditional expression is a series of relations joined together with AND and OR operators. Parentheses () are also supported. The basic relation compares an attribute name with a value through a relational operator.

For example,

```
is_hierarchical == true and area <= 6
```

If an attribute is a collection attribute that contains a single object, then you can query an attribute value from that object using "." to name the attribute on the other object. The "." operator can be chained. For example.

```
owner.is_hierarchical == true and owner.area <= 6
shape.net.name==reset
```

The value side of a relation can be a simple string, quoted string, or an attribute name prefixed with "@". For example:

```
input_delay<=@output_delay
input_delay<=0.24
name=="@literal_name"
```

The relational operators are

```
== Equal
!= Not equal
> Greater than
< Less than
>= Greater than or equal to
<= Less than or equal to
=~ Matches pattern
!~ Does not match pattern
```

The basic relational rules are

- String attributes can be compared with any operator.
- Numeric attributes cannot be compared with pattern match operators.
- Boolean attributes can be compared only with == and !=. The value can be only either *true* or *false*.

Existence relations determine if an attribute is defined or not defined for the object. For example,

```
(sense == setup_clk_rise) and defined(sdf_cond)
```

The existence operators are

```
defined
undefined
!defined
```

These operators apply to any attribute as long as it is valid for the object class.

Collection attributes support a special sizeof(attrName) operator that returns the number of objects in the attribute. If the attribute is not set then 0 is returned.

The **filter_collection** command has a *-regex* option that uses regular expressions when matching for string attributes. Regular

expression matching is done in the same way as in the Tcl **regexp** command. When using the *-regexp* option, take care in the way you quote the filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended.

Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding *".*"* to the beginning or end of the expressions as needed. You can make the regular expression search case-insensitive by using the *-nocase* option.

If the application variable `filter_collection_extended_syntax` is true then mathematical expressions are supported too. All function names must be prefixed with the '#' in order to disambiguate the function names from subscripted attribute names (i.e. `"#sin(attr)"`). In order to use a math expression on the right hand side of a comparison, the expression must be contained in parenthesis. This is required for backward compatibility with the legacy expression syntax as strings on the right do not require quoting.

When extended syntax is on filters can also match against glob style subscript patterns (i.e. `"input_delay(*)<1.0"`). In that case the term evaluates to true if any subscript satisfies the condition.

Additionally when extended syntax is on, a filter may refer to nested attributes that may imply multiple objects in a collection (i.e. `"pins.capacitance==0.2"`). In that case the term returns true if any of the objects referred to in the have an attribute value satisfying the condition.

EXAMPLES

The following example from PrimeTime creates a collection of only hierarchical cells.

```
prompt> set a [filter_collection [get_cells *] \
  "is_hierarchical == true"]
{Adder1 Adder2}
```

The following example from PrimeTime uses existence operators to create a collection of all nonmoded cell timing_arc objects in the current design.

```
prompt> set b [filter_collection \
  [get_timing_arcs -of_objects [get_cells *]] \
  "undefined(mode)"]
```

The following example finds all cells from {U0(1) U1(1) U0(0) U1(0)} with the index (0). Notice the round brackets used for the index.

```
prompt> filter_collection -regexp [get_cells] {full_name =~ "U.*\0\0"}
{U0(0) U1(0)}
```

The following example finds all ports from in[18] to in[23]. The placement of double quotes and backslashes is necessary to avoid syntax errors.

```
prompt> filter_collection [get_ports] -regexp {name =~ "in\[1[8-9]2[0-3]\]\\"}
{in[23] in[22] in[21] in[20] in[19] in[18]}
```

The following example finds the U2 and U4 cells, using the *-nocase* option:

```
prompt> filter_collection [get_cells] -regexp -nocase {name =~ u[24]}
{U2 U4}
```

SEE ALSO

```
collections(2)  
regexp(2)  
filter_collection_extended_syntax(3)
```

find

Finds a design or library object.

SYNTAX

```
list find
  type
  [name_list]
  [-hierarchy]
  [-flat]
```

Data Types

```
type      one-of-string
name_list list
```

ARGUMENTS

type

Specifies the type of object to find. The value of *type* can be any of the following:

design	clock
port	reference
cell	net
pin	cluster
rp_group	library
lib_cell	lib_pin
multibit	operator
module	implementation
file	

The value can also be **physnet**, **physcell**, or **physport** for physical objects.

name_list

Specifies a list of names of design or library objects in dc_shell to find. If you do not specify *name_list*, all objects of the specified type are returned.

-hierarchy

Specifies to return all objects matching *type* and *name_list* within the current design hierarchy. The full name of the object at a particular level must match the patterns. The search is similar to the UNIX **find** command. For example, if there is a cell named

block1/adder, a hierarchical search finds it using *adder*. If you use the **-hierarchy** option, the *type* must be either **design**, **lib_cell**, **net**, **cell**, or **pin**.

-flat

Specifies that the command finds only objects in leaf cells. You must specify **-flat** with the **-hierarchy** option.

DESCRIPTION

The **find** command locates and returns a list of design or library objects in *dc_shell*. The command returns all of the design or library objects of the given *type* that match the names specified in the *name_list*. If no objects are found, the return value is an empty list (*{}*).

In Tcl-based *dc_shell*, the **find** command returns a collection.

The **find** command is generally used as a parameter to another *dc_shell* command. In this case, the parameters of **find** must be enclosed in parentheses to separate them from the parameters of the other command.

The **find** command is used to resolve name conflicts for commands that accept objects of more than one type. For example, the **set_dont_touch** command accepts design cells, nets, references, clusters, and library cells as arguments. If the current design contains both a net and cell named *interrupt* the name *interrupt* does not specify a unique object. Depending on the type of object expected, *interrupt* can mean the net interrupt or the cell interrupt. Resolve this explicitly with the following command:

```
prompt> set_dont_touch find (net, interrupt)
```

Naming Conventions

A consistent naming convention is used to refer to design and library objects by all of the commands in *dc_shell*. In most cases, the referenced object exists in the current design and can be identified by using its simple name. More complex cases can be handled by including a library specification prefix. This prefix allows you to reference objects in any library and objects in the current design. The design specification prefix is separated from the simple object name with a / (slash).

In the following example, the *MUX51HP* cell in the *Isi_10k* library is returned:

```
prompt> find cell Isi_10k/MUX51HP
{A}
```

You can uniquely identify any design or library object in *dc_shell* by also specifying the design file in the design specification prefix. This portion of the name is delimited by a : (colon). Do this only when you have two designs (or libraries) with the same name that originated from different file names, as in the following example:

```
prompt> find port /project/ADDER.db:ADDER/A
{A}
```

Objects are subordinate to a design or library object. The types of objects that are contained in a design object are ports, references, clocks, cells, nets, pins, clusters, and Relative Placement (RP) groups. The types of objects contained in a library object are *lib_cells*, modules, implementations, and pins.

To find an object that might exist in a design or a library, a search order is implied. The following example shows the search procedure that **find** uses to locate objects:

```
prompt> find cell ADDER/A
```

To determine the object that **find** returns depends on a two-step search:

1. Search the current design for a cell named *ADDER/A*. If one exists, return it.
2. Extract the library specification and search for the library *ADDER*. Search the *ADDER* library for a cell named *A*. If one exists,

return that cell name. Otherwise, return an empty list.

You can use the wildcard character `*` (asterisk) to match any number of characters in a name, or you can use `?` (question mark) to match a single character. For example, `u*` specifies all object names that begin with the letter `u`. Specifying `u*z` returns all object names that begin with the letter `u` and end with the letter `z`. Specifying `u?v` returns all object names that begin with the letter `u` and end with letter `v` matching any single character in the middle.

Wildcard characters must be escaped using double backslashes (`\\`) to remove their special regular expression meaning. For example, `u\\*z` specifies any object with the name `u*z`. For more details about escaping wildcards refer to the **wildcards** variable man page.

Implicit Find Command

Commands that accept design or library object arguments perform an implicit **find** for each argument that is not already an explicit find. For example, the **set_input_delay** command accepts only port objects. Therefore, the following two commands are equivalent:

```
prompt> set_input_delay IN1 10
```

```
prompt> set_input_delay find (port, IN1) 10
```

The implicit **find** operation is different for commands that accept more than one object type. In this case, each object type is searched (in order) for a match to the name. The search ends when a match is found. For example, the following command first looks for a cell named `X`, then a net named `X`, next a reference named `X`, and finally a library cell named `X`:

```
prompt> set_dont_touch X
```

The implicit **find** operation is different for commands that operate only on bus objects. In this case, each object type is searched (in order) for a match to the name. It first searches ports, and then searches nets. The search of object types ends when a match is found. If one of the objects found is a bus object, it is among the objects returned in the list. If one of the objects found is not a bus object, it is among the objects returned only if it is not a member of a bus object. For example, the following command first looks for ports with names that begin with the letter `X`, then for nets with names that begin with the letter `X`:

```
prompt> remove_bus X*
```

If there are ports named `XJ1`, `XJ2`, `XK1`, `XK2`, `XL1`, and `XL2`, and there are port buses named `XJ` (with members `XJ1` and `XJ2`) and `XK` (with members `XK1` and `XK2`), the objects returned are ports `XL1`, and `XL2`, and the port buses `XJ` and `XK`. Nets are not searched for because ports are found.

Hierarchical Find Command

The **-hierarchy** option to **find** returns all objects within the hierarchy of the current design that meet the *type* and *name_list* criteria.

The impact of each allowed *type* used in conjunction with the **-hierarchy** option is described as follows:

find design -hierarchy

Returns all design objects referenced within the hierarchy of the current design.

find lib_cell -hierarchy

Returns all `lib_cell` objects referenced within the hierarchy of the current design.

find net -hierarchy

Returns all instances of nets within the hierarchy of the current design.

find cell

Returns all instances of cells within the hierarchy of the current design.

find pin

Returns all instances of pins within the hierarchy of the current design.

Adding a *name_list* in each of the previous cases further restricts the returned objects to a given name.

EXAMPLES

The following example finds the *N1* net in the current design:

```
prompt> find net N1
      {N1}
```

The following example uses the * (asterisk) wildcard character to find all cells in the current design that begin with *U*:

```
prompt> find cell U*
      {U0, U1, U2, U3, U4, U5, U6, U7}
```

The following example finds the *CACHE/MEM* cluster in the current design:

```
prompt> find cluster CACHE/MEM
      {MEM}
```

The following example uses a library prefix to find the *IVA* cell in the *tech_lib* library and sets the **dont_use** attribute:

```
prompt> set_dont_use tech_lib/IVA
```

The following example finds all pins of the *U8* cell:

```
prompt> find pin U8/*
      {U8/A, U8/Z}
```

The following example sets the **dont_touch** attribute for design reference *adder* of the current design:

```
prompt> set_dont_touch find (reference, adder)
```

The following example sets the **dont_touch** attribute for *datapath* cluster of the current design:

```
prompt> set_dont_touch find (cluster, datapath)
```

In the following example, the specified object is not found and a warning is returned:

```
prompt> find port A1
Warning: Can't find port 'A1' in design 'adder'
      {}
```

If you specify multiple names, **find** returns all of the objects found, and a warning is returned for those not found:

```
prompt> find design {add, hadd, fadd}
Warning: Can't find design 'hadd'
      {add, fadd}
```

The following example shows how to prefix a name with a full file specification:

```
prompt> find cell /project/ripple/lookahead/TOP.db:MULTX
      {MULTX}
```

The following examples show the use of **find** where *type* has the value of **file**:

```
prompt> list -files
File      Path
----      -
adder.db   /remote/usr4/joeuser/designs
clock.db   /remote/usr4/joeuser/designs
1
```

```
prompt> find file "adder.db"
{adder.db}
```

```
prompt> find file "mux.db"
Error: Can't find file 'mux.db'. (UID-109)
{}
```

```
prompt> remove_design find(file,"clock.db")
Removing file 'clock.db'
design 'TOP'
design 'MUX'
design 'TIME_BLOCK'
design 'TIME_STATE_MACHINE'
design 'TIME_COUNTER'
design 'TIME_COUNTER_DW01_inc_n6_0'
design 'ALARM_BLOCK'
design 'ALARM_COUNTER'
design 'ALARM_COUNTER_DW01_inc_n6_0'
design 'ALARM_STATE_MACHINE'
design 'ALARM_SM_2'
design 'CONVERTOR_CKT'
design 'CONVERTOR'
design 'HOURS_FILTER'
design 'COMPARATOR'
1
```

If you specify **-hierarchy** using **design** as the value of **type**, all designs referenced within the current design are returned as shown in the following example:

```
prompt> find design -hierarchy
{HALF_SUBTRACTOR FULL_SUBTRACTOR SUBTRACTOR}
```

The following example returns all instances of cells that begin with the letter *U* in the hierarchy of the current design:

```
prompt> find -hierarchy cell "U*"
{U1MY_CELL/U1MY_CELL/U2U1/EXAMPLE/U33}
```

The following example returns all instances of leaf cells that begin with *U1* in the hierarchy of the current design:

```
prompt> find -flat -hierarchy cell "U1*"
{U4/U10 U4/U11 U4/U12 U1/U2/U132 U1/U2/U131 U6/U10}
```

The following example returns all multibit components that begin with *U1* in the current design:

```
prompt> find multibit "U*"
{U2 U4_multibit}
```

The following example in Tcl mode returns and displays a collection using the implicit query property:

```
prompt> find cell U*  
      {U1 U2 U345}
```

The following example in Tcl mode sets variable *a* to contain a collection. The implicit query property of **find** in Tcl mode returns the collection, even though the name assigned to the collection is *-se11*, as shown by using the **echo** command:

```
prompt> set a [find cell U*]  
      {U1 U2 U345}  
prompt> echo $a  
      "_se11"
```

SEE ALSO

- collections(2)
- current_design(2)
- list(2)
- get_attribute(2)
- get_cells(2)
- get_clocks(2)
- get_clusters(2)
- get_designs(2)
- get_generated_clocks(2)
- get_lib_cells(2)
- get_lib_pins(2)
- get_libs(2)
- get_multibits(2)
- get_nets(2)
- get_pins(2)
- get_ports(2)
- get_references(2)
- remove_attribute(2)
- report_lib(2)
- set_attribute(2)
- attributes(3)
- find_allow_only_non_hier_ports(3)
- find_converts_name_lists(3)
- wildcards(3)

find_objects

Finds logical hierarchy objects within a scope. This is a UPF query command.

SYNTAX

```
list find_objects
  scope
  -pattern pattern_string
  -object_type inst | port | net | model | supply_port | process
  [-direction in | out | inout]
  [-transitive true | false]
  [-non_leaf]
  [-leaf_only]
  [-traverse_macros]
  [-ignore_case]
  [-regexp | -exact]
```

Data Types

```
scope          string
pattern_string string
```

ARGUMENTS

scope

Specifies the scope within in to search for logical hierarchy objects.

-pattern *pattern_string*

Specifies a search pattern. By default, the pattern is treated as a Tcl glob expression.

-object_type inst | port | net | model | supply_port | process

Searches only for the specified object type.

The keyword values have the following definitions:

- **inst** (cell instance)
- **port** (design port)
- **net** (net)
- **model** (model or lib cell)

- **supply_port** (UPF supply port or P/G pin)
- **process** (VHDL process)

If **object_type model** is specified, **find_objects** searches for instances of any model and lib cell whose name matches the *search_pattern*. If **object_type** is specified with any other value then **find_objects** searches the logic hierarchy for the specified objects whose name matches the *search_pattern*.

-direction in | out | inout

Restricts the direction of the searched ports.

This option has an effect only when ports are included in the search.

-transitive true | false

Controls whether the descendants of the elements are included in the search.

By default (**false**), the descendants are not included in the search.

-non_leaf

Limits the search to only nonleaf design elements (elements that have child elements).

If you do not specify the **-non_leaf** or **-leaf_only** option, the command searches for both leaf and nonleaf design elements.

-leaf_only

Limits the search to only leaf-level design elements (elements without child elements).

If you do not specify the **-non_leaf** or **-leaf_only** option, both leaf and nonleaf design elements are searched.

-traverse_macros

By default, during a search, the tool will not descend past the boundary of hierarchy marked as "**terminal boundary**" or as "**soft macro**". The **-traverse_macros** option can be used to override the default behavior, allowing the tool to continue searching into said hierarchy.

When the **-traverse_macros** option is used, the search will be performed as if the **-transitive** option was set to TRUE.

-ignore_case

Performs case-insensitive pattern matching. By default, all pattern matches are case sensitive.

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the `=~` and `!~` filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding `".*"` to the beginning or end of the expressions, as needed.

The **-regexp** and **-exact** options are mutually exclusive. You can specify only one of these options.

-exact

Disallows wildcard expansion on the specified *search_pattern*. When this option is not used, *search_pattern* is interpreted as a Tcl glob expression.

The `-regex` and `-exact` options are mutually exclusive. You can specify only one of these options.

DESCRIPTION

The **find_objects** command finds logical hierarchy objects within a scope. It returns a list of the found hierarchical names relative to the active scope. When nothing is found, it returns a null string.

This is a UPF query command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

Here are some examples of usage of the command with different options.

```
prompt> find_objects a -pattern * -object_type inst \  
-transitive TRUE -non_leaf  
a/b1 a/b2
```

```
prompt> find_objects a/b2 -pattern b_? -object_type net  
a/b2/b_i a/b2/b_o
```

```
prompt> find_objects a/b1 -pattern * -object_type port \  
-transitive TRUE -direction out  
a/b1/c1/Z a/b1/c2/Z a/b1/b_o
```

```
prompt> find_objects . -object_type model -pattern b* -non_leaf \  
-transitive TRUE  
a/b1 a/b2 b1
```

```
prompt> find_objects . -object_type model -pattern b* -non_leaf \  
-transitive TRUE -exact  
Warning: Can't find module or lib cell 'b*' in design 'top'. (UID-95)
```

```
prompt> find_objects . -object_type model -pattern bot -non_leaf \  
-transitive TRUE -exact  
a/b1 a/b2 b1
```

SEE ALSO

get_nets(2)
get_ports(2)
get_cells(2)


```
get_designs(2)  
set_scope(2)
```

foreach

Specifies the control structure for list traversal loop execution.

SYNTAX

status **foreach**

ARGUMENTS

The **foreach** command has no arguments.

DESCRIPTION

This command iteratively executes commands while successively setting the *variable_name* argument to each member of *list_expression*. Note that this changes the type of the *variable_name* argument for each iteration of the loop.

The *list_expression* argument is evaluated upon execution and converted to a list-type expression, as appropriate. Commands in *loop_statement_block* are executed once for each member in the list that results from evaluating the *list_expression* argument. On each loop iteration, the *variable_name* argument is set to each successive member in the list.

The variable specified by the *variable_name* argument and its value remain after the loop terminates. If the **foreach** command is used within a user-defined function, the *variable_name* variable is local to that function.

The return value of the **foreach** command is the return value of the last command executed. If no commands execute, the return value is 0.

The **break** and **continue** commands modify the standard execution of the **foreach** command. See the **break** and **continue** command man pages for more information.

You can redirect the output of the **foreach** command to a file. This file contains the output of any commands executed in the body of the **foreach** command.

EXAMPLES

The following example uses the **foreach** command to print the elements of a simple list:

```
prompt> set mylist {I1/FF3/D I1/FF4/D I1/FF5/D}
```

```
I1/FF3/D I1/FF4/D I1/FF5/D
```

```
prompt> foreach i $mylist {echo $i}  
I1/FF3/D  
I1/FF4/D  
I1/FF5/D
```

SEE ALSO

break(2)
continue(2)
if(2)
while(2)

foreach_in_collection

Iterates over the elements of a collection.

SYNTAX

```
string foreach_in_collection  
    itr_var  
    collections  
    body
```

Data Types

<i>itr_var</i>	string
<i>collections</i>	list
<i>body</i>	string

ARGUMENTS

itr_var

Specifies the name of the iterator variable.

collections

Specifies a list of collections over which to iterate.

body

Specifies a script to execute per iteration.

DESCRIPTION

The **foreach_in_collection** command is used to iterate over each element in a collection. You cannot use the Tcl-supplied **foreach** command to iterate over collections because the **foreach** command requires a list, and a collection is not a list. Also, using the **foreach** command on a collection causes the collection to be deleted.

The arguments for the **foreach_in_collection** command parallel those of the **foreach** command: an iterator variable, the collections over which to iterate, and the script to apply at each iteration. All arguments are required.

Note: The **foreach_in_collection** command does not allow a list of iterator variables.

During each iteration, the *itr_var* option is set to a collection of exactly one object. Any command that accepts *collections* as an

argument also accepts *itr_var* because they are of the same data type (collection).

You can nest the **foreach_in_collection** command within other control structures, including another **foreach_in_collection** command.

Note that if the body of the iteration is modifying the netlist, it is possible that all or part of the collection involved in the iteration will be deleted. The **foreach_in_collection** command is safe for such operations. If a command in the body causes the collection to be removed, at the next iteration, the iteration ends with a message indicating that the iteration ended prematurely.

An alternative to collection iteration is to use complex filtering to create a collection that includes only the desired elements, then apply one or more commands to that collection. If the order of operations does not matter, the following are equivalent. The first is an example without iterators.

```
set s [get_cells {U1/*}]
command1 $s
command2 $s
unset s
```

The following is a similar approach using the **foreach_in_collection** command:

```
foreach_in_collection itr [get_cells {U1/*}] {
  command1 $itr
  command2 $itr
}
```

For collections with large numbers of objects, the non-iterator version is more efficient, though both produce the same results if the commands are order-independent.

EXAMPLES

The following example from PrimeTime removes the wire load model from all hierarchical cells in the current instance.

```
prompt> foreach_in_collection itr [get_cells *] {
?      if {[get_attribute $itr is_hierarchical] == "true"} {
?          remove_wire_load_model $itr
?      }
?      }
Removing wire load model from cell 'i1'.
Removing wire load model from cell 'i2'.
```

SEE ALSO

collections(2)
foreach(2)

generate_mv_constraints

Generates new isolation strategies for design elements.

SYNTAX

```
status generate_mv_constraints
  [-align_isolation_clamp_value]
  [-no_isolation]
  [-output output_file_name]
  [-include_elements]
  [-apply]
  [-elements element_list]
```

Data Types

```
output_file_name  string
element_list      list
```

ARGUMENTS

-align_isolation_clamp_value

Optional Argument. Enables the **generate_mv_constraints** command to try to align the isolation clamp value for the constant elements.

-no_isolation

Optional Argument. Enables the **generate_mv_constraints** command to create no isolation strategies for elements where redundant isolation cells would be inserted.

-output *output_file_name*

Optional argument which tells **generate_mv_constraints** where to write out the newly created strategies. If the file specified exists, it is overwritten.

-include_elements

Optional argument, used in conjunction with the **-align_isolation_clamp_value** option. By default, the **generate_mv_constraints** command does not attempt to correct any elements specified in the **-elements** option of the **set_isolation** command to align with the isolation clamp value. When specified, the tool modifies the user-defined isolation strategies having **-elements** by removing any constant elements not matching the isolation clamp value and adding them to the tool derived isolation strategies with the clamp value aligned to the constant value of the element. It is an optional argument.

-apply

If specified, applies all the newly created strategies to the design in memory. It is an optional argument.

-elements *element_list*

list of design objects that are going to be processed by the command. Accepts pins, ports and cells references.

DESCRIPTION

The purpose of this utility is to help refine the user UPF before inserting isolation cells.

The **-no_isolation** option helps reduce the number of redundant isolation cell that would be inserted by assigning **-no_isolation** isolation strategies to design elements where such redundant isolation cells would be inserted.

The **-elements** option is required when using the **-no_isolation** option and cannot be used without it

The **-align_isolation_clamp_value** option analyzes the isolation strategies and derives new isolation strategies for the elements whose driving constant value is different than the specified isolation clamp value.

The **-align_isolation_clamp_value** option must be used before isolation cell insertion takes place, otherwise back-to-back isolation cell might appear and/or previously associated isolation cells might become unassociated.

The **-align_isolation_clamp_value** option does not align the clamp value of elements where the current applying isolation strategy that was created using the **-no_isolation** (from the **set_isolation** command) or the **-force_isolation** option or with a **-clamp_value** other than 0 or 1.

The **-align_isolation_clamp_value** option and the **-no_isolation** option are mutually exclusive.

generate_mv_constraints requires the design and UPF data to be loaded.

The tool derived strategies are written to the output file specified with the **-output** option. In addition to the **set_isolation** constraint, the utility also emits the **set_isolation_control** constraint for these new strategies. However, no **map_isolation_cell** constraint are emitted and the **UPF-442** message is printed. You need to add a relevant **map_isolation_cell** constraint.

The tool derived strategies can also be applied to the netlist in memory when **-apply** option is specified.

EXAMPLES

Following examples show different usages of **generate_mv_constraints** command and the outcome.

```
prompt> generate_mv_constraints
Error: Required argument '-align_isolation_clamp_value' was not found. (CMD-007)
```

```
prompt> generate_mv_constraints -align_isolation_clamp_value
Error: Either '-output' or '-apply' needs to be specified. (UPF-445)
```

Following shows the behavior when **generate_mv_constraints** command is called without loading any UPF constraint.

```
prompt> generate_mv_constraints -align_isolation_clamp_value
Error: Isolation clamp value is not aligned, as there is no UPF data. (UPF-440)
```

Following shows the behavior when **generate_mv_constraints** command identifies a constant with non-matching clamp value.

The user-provided UPF has the following isolation constraint where element u1/cin is driven by constant 0.

```
prompt> set_isolation iso1 \
```

```

-domain CARRY -applies_to inputs \
-isolation_power_net VDD -isolation_ground_net GND \
-clamp_value 1
prompt> set_isolation_control iso1 \
-domain CARRY -isolation_signal ctrl \
-location self -isolation_sense low

prompt> generate_mv_constraints -align_isolation_clamp_value \
-output align.upf
1

```

The output file align.upf is as follows:

```

# Created by DC Utility for non-matching isolation clamp and driving constant value.

# List of new strategies created with clamp value matching the constant isolating it, for strategies with no user specified element list
set_isolation iso1_clamp0 -domain CARRY -isolation_power_net VDD -isolation_ground_net GND -elements u1/cin -clamp_value
set_isolation_control iso1_clamp0 -domain CARRY -isolation_signal ctrl -isolation_sense low -location self

```

Following shows the behavior when **generate_mv_constraints** command identifies a constant with nonmatching clamp value.

The user-supplied UPF has the following isolation constraint where element u1/cin is driven by constant 0.

```

prompt> set_isolation iso1 \
-domain CARRY -applies_to inputs \
-elements {u1/cin u1/carry} -clamp_value 1
prompt> set_isolation_control iso1 \
-domain CARRY -isolation_signal ctrl \
-location self -isolation_sense low

prompt> generate_mv_constraints -align_isolation_clamp_value \
-output align.upf -include_elements
1

```

The output file align.upf is as follows:

```

# Created by DC Utility for non-matching isolation clamp and driving constant value.

# List of user strategies modified
set_isolation iso1_modified -domain CARRY -isolation_power_net VDD -isolation_ground_net GND -elements u1/carry -clamp_value
set_isolation_control iso1_modified -domain CARRY -isolation_signal ctrl -isolation_sense low -location self

# List of new strategies created with clamp value matching the constant isolating it, for strategies with user specified element list
set_isolation iso1_clamp0 -domain CARRY -isolation_power_net VDD -isolation_ground_net GND -elements u1/cin -clamp_value
set_isolation_control iso1_clamp0 -domain CARRY -isolation_signal ctrl -isolation_sense low -location self

```

SEE ALSO

```

set_isolation(2)
set_isolation_control(2)
map_isolation_cell(2)

```

generate_rtl_upf

Writes out the design's UPF power intent as a UPF command script in DC Explorer.

SYNTAX

```
status generate_rtl_upf  
-path path_name
```

Data Types

```
path_name    string
```

ARGUMENTS

DESCRIPTION

The **generate_rtl_upf** command creates a modified version of the input UPF file from the user. User-specified UPF commands that are not supported by DC Explorer are commented out, and UPF commands derived by the tool for auto-isolation insertion, such as **set_isolation**, **set_isolation_control**, **create_pst**, **add_pst_state**, and **add_power_state**, are added. The output UPF file reflects the power intent of the design as seen by DC Explorer for the **compile_exploration** command.

The **generate_rtl_upf** command should be called immediately after all of the user's UPF power intent has been loaded.

The tool appends the .de_upf_rtl suffix to the original UPF file names as the output file names generated by the **generate_rtl_upf** command.

EXAMPLES

The following example saves the full UPF power intent of the design in a file in the current directory. The resulting UPF file name is top.upf.de_upf_rtl.

```
prompt> load_upf top.upf  
prompt> generate_rtl_upf -path ./
```

SEE ALSO

load_upf(2)
save_upf(2)

get_alternative_lib_cells

Creates a collection of equivalent library cells for a specified cell or library cell.

SYNTAX

```
collection get_alternative_lib_cells
  [-quiet]
  [-regexp | -exact]
  [-nocase]
  [-filter expression]
  [-libraries libraries]
  pattern_or_objects
```

Data Types

```
expression      string
libraries       string
pattern_or_objects string
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *pattern_or_objects* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding "." to the beginning or end of the expressions, as needed.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive, both for the *patterns_or_objects* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each library cell in the collection, the expression is evaluated based on the library cell's attributes. If the expression evaluates to true, the library cell is included in the result.

To see the list of library cell attributes that you can use in the expression, use the **list_attributes -application -class lib_cell** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-libraries *libraries*

Specifies a list of libraries from which to select equivalent library cells. You can specify a list of library names or a collection of libraries loaded into memory.

If you do not specify this option, the command traverses the link path to find libraries. In this case, if a library is found with a minimum library relationship (which is determined from the setting in the **set_min_library** command), the minimum library is automatically included.

pattern_or_objects

Specifies the cells or library cells for which to return equivalent library cells.

Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

This is a required option.

DESCRIPTION

This command creates a collection of library cells that are logically similar to the specified cell or library cell and pass the filter, if specified. This collection can be used to replace or resize the specified cell in the current design.

If the command cannot find any logically similar library cells for the specified cell and the current design is not linked, the design automatically links.

The command returns a collection if any library cells match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the equivalent library cells for the slow/AND2X2 library cell.

```
prompt> set libcsel [get_lib_cells slow/AND2X2]
{slow/AND2X2}
```

```
prompt> [get_alternative_lib_cells $libcsel]
{slow/AND2X1 slow/AND2X6 slow/AND2X12}
```

The following example queries the equivalent library cells for the top/U21 cell instance.

```
prompt> set csel [get_cell top/U21]
{top/U21}
```

```
prompt> [get_alternative_lib_cells $csel]
{slow/AND2X1 slow/AND2X2 slow/AND2X6 slow/AND2X12}
```

SEE ALSO

- collections(2)
- filter_collection(2)
- get_cells(2)
- get_lib_cells(2)
- list_attributes(2)
- query_objects(2)
- set_min_library(2)
- collection_result_display_limit(3)
- wildcards(3)

get_always_on_logic

Returns a collection of cells and nets on the always-on paths in the design.

SYNTAX

```
collection get_always_on_logic
  [-cells]
  [-nets]
  [-all]
  [-boundary]
```

ARGUMENTS

-cells

Returns the buffers and inverters on the always-on paths.

-nets

Returns the nets on the always-on paths.

-all

Returns the nets, buffers, and inverters on the always-on paths. This is the default behavior when no argument is specified.

-boundary

Returns the isolation and level-shifter cells on the boundary of the always-on paths.

DESCRIPTION

This command creates a collection of nets and cells on the always-on paths. The always-on paths are paths that must always be powered up during the operation of the system. The always-on paths consist of logic cones that terminate on always-on pins in the design. Examples of always-on pins are the enable pin of isolation cells and special control pins of retention registers.

You can mark any pin of a cell instance with an always-on attribute by using the **set_attribute** command. You can create power-down regions by using the **create_power_domain** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns the always-on nets, buffers, and inverters in the design.

```
prompt> get_always_on_logic
```

The following example returns the always-on nets, buffers, inverters, boundary level-shifter cells, and isolation cells in the design.

```
prompt> get_always_on_logic -boundary
```

The following example returns the always-on nets, boundary level-shifter cells and isolation cells in the design.

```
prompt> get_always_on_logic -nets -boundary
```

The following example returns the always-on buffers, inverters, boundary level-shifter cells, and isolation cells in the design.

```
prompt> get_always_on_logic -cells -boundary
```

The following example returns the always-on nets, buffers, inverters, boundary level-shifter cells, and isolation cells in the design.

```
prompt> get_always_on_logic -nets -cells -boundary
```

SEE ALSO

`create_power_domain(2)`
`set_attribute(2)`

get_app_var

Gets the value of an application variable.

SYNTAX

```
string get_app_var  
  [-default | -details | -list]  
  [-only_changed_vars]  
  var
```

Data Types

var string

ARGUMENTS

-default

Gets the default value.

-details

Gets additional variable information.

-list

Returns a list of variables matching the pattern. When this option is used, then the *var* argument is interpreted as a pattern instead of a variable name.

-only_changed_vars

Returns only the variables matching the pattern that are not set to their default values, when specified with **-list**.

var

Specifies the application variable to get.

DESCRIPTION

The **get_app_var** command returns the value of an application variable.

There are four legal forms for this command:

- `get_app_var <var>`
Returns the current value of the variable.
- `get_app_var <var> -default`
Returns the default value of the variable.
- `get_app_var <var> -details`

Returns more detailed information about the variable. See below for details.

- `get_app_var -list [-only_changed_vars] <pattern>`

Returns a list of variables matching the pattern. If *-only_changed_vars* is specified, then only variables that are changed from their default values are returned.

In all cases, if the specified variable is not an application variable, then a Tcl error is returned, unless the application variable **sh_allow_tcl_with_set_app_var** is set to true. See the **sh_allow_tcl_with_set_app_var** man page for details.

When **-details** is specified, the return value is a Tcl list that is suitable as input to the Tcl **array set** command. The returned value is a list with an even number of arguments. Each odd-numbered element in the list is a key, and each even-numbered element in the list is the value of the previous key.

The supported keys are as follows:

name

This key contains the name of the variable. This key is always present.

value

This key contains the current value of the variable. This key is always present.

default

This key contains the default value of the variable. This key is always present.

help

This key contains the help string for the variable. This key is always present, but sometimes the value is empty.

type

This key contains the type of the application variable. Legal values of for this key are: string, bool, int, real. This key is always present.

constraint

This key describes additional constraints placed on this variable. Legal values for this key are: none, list, range. This key is always present.

min

This key contains the min value of the application variable. This key is present if the constraint is range. The value of this key may be the empty string, in which case the variable only has a max value constraint.

max

This key contains the max value of the application variable. This key is present if the constraint is "range". The value of this key may be the empty string, in which case the variable only has a min value constraint.

list

This key contains the list of legal values for the application variable. This key is present if the constraint is "list".

EXAMPLES

The following are examples of the **get_app_var** command:

```
prompt> get_app_var sh_enable_page_mode  
1
```

```
prompt> get_app_var sh_enable_page_mode -default  
false
```

```
foreach {key val} [get_app_var sh_enable_page_mode -details] { echo "$key: $val"  
}  
=> name: sh_enable_page_mode  
    value: 1  
    default: false  
    help: Displays long reports one page at a time  
    type: bool  
    constraint: none
```

```
prompt> get_app_var -list sh_*message  
sh_new_variable_message
```

SEE ALSO

```
report_app_var(2)  
set_app_var(2)  
write_app_var(2)
```

get_attribute

Returns the value of an attribute on a list of design or library objects.

SYNTAX

```
list get_attribute  
    object_list  
    attribute_name  
    [-bus]  
    [-quiet]  
    [-return_null_values]
```

Data Types

```
object_list    list  
attribute_name string
```

ARGUMENTS

object_list

Specifies a list of the design or library objects for which the attribute value is returned.

attribute_name

Specifies the name of the attribute whose value is returned.

-bus

Returns the value of the attribute that is on the bus as a whole, and not on each individual bus member.

-quiet

Turns off the warning message that is otherwise issued if the attribute or objects are not found.

-return_null_values

Returns an empty string for failed objects when the *object_list* argument contains multiple objects.

DESCRIPTION

This command searches the *object_list* for the specified attribute and returns a list of attribute values. By default, if a specified object

cannot be found or the attribute does not exist on the object, that object is ignored. If you use the **-return_null_values** option, the command returns an empty string.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example gets the value of the **rise_drive** attribute for the IN1 port.

```
prompt> get_attribute [get_ports IN1] rise_drive
0.25
```

The following example gets the value of the **load** attribute for all ports whose names begin with OUT:

```
prompt> get_attribute get_ports OUT* load
1.0 2.0 2.5 1.0
```

The following example gets the value of the **area** attribute for the G1 and G2 cells in the tech_lib library:

```
prompt> get_attribute {tech_lib/G1 tech_lib/G2 } area
1.0 2.0
```

The following example sets a user-defined attribute named X on some cells and then uses **get_attribute** commands to retrieve this attribute value, as well as the value of an application attribute.

```
prompt> define_user_attribute -type int -class cell X
X
prompt> set_attribute [get_cells *] X 30
Information: Attribute 'X' is set on 2 objects. (UID-186)
i1 i2
prompt> foreach_in_collection sel [get_cells *] {
echo -n "On '[get_attribute $sel full_name]', "
echo "X = [get_attribute $sel X]"
}
On 'i1', X = 30
On 'i2', X = 30
```

In the following example, the **-return_null_values** option returns an empty string when the tool cannot find the attributes for some objects.

```
prompt> get_attribute [get_cells {cell1 cell2 cell3}] area
Warning: Attribute 'area' does not exist on cell 'cell2' (ATTR-3)
79.833600 79.833600
```

```
prompt> get_attribute [get_cells {cell1 cell2 cell3}] area \
-return_null_values
Warning: Attribute 'area' does not exist on cell 'cell2' (ATTR-3)
79.833600 {} 79.833600
```

The following example shows the warning that is issued because the U9 cell is not found:

```
prompt> get_attribute [get_cells {U1 U9}] dont_touch
Warning: Can't find object 'U9' in design 'example'. (UID-95)
```

true
-0.25in

SEE ALSO

collections(2)
define_user_attribute(2)
foreach_in_collection(2)
list_attributes(2)
remove_attribute(2)
set_attribute(2)
attributes(3)

get_bounds

Creates a collection of bounds from the current design.

SYNTAX

```
collection get_bounds
  [-quiet]
  [-regex | -exact]
  [-nocase]
  [-filter expression]
  [patterns | -of_objects objects]
```

Data Types

```
expression  string
patterns    list
objects     list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Does not suppress syntax error messages.

-regex

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the == and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regex** command. When using the **-regex** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly brackets around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each bound in the collection, the expression is evaluated based on the bound's attributes. If the expression evaluates to true, the bound is included in the result.

To see the list of bound attributes that you can use in the expression, use the **list_attributes -application -class bound** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

patterns

Creates a collection of bounds whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

The *patterns* argument and **-of_objects** option are mutually exclusive; you can use only one. If you do not use either one, the command uses the asterisk (*) as the default pattern.

-of_objects *objects*

Creates a collection of bounds that contain the specified objects. The objects must be leaf cells or ports.

The *patterns* argument and **-of_objects** option are mutually exclusive; you can use only one. If you do not use either one, the command uses the asterisk (*) as the default pattern.

DESCRIPTION

This command creates a collection of bounds from the current design that match the specified criteria.

The command returns a collection if any bounds match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects** or **report_bounds**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets all bounds with names that start with my_bound.

```
prompt> get_bounds my_bound*
```

SEE ALSO

- `collections(2)`
- `filter_collection(2)`
- `foreach_in_collection(2)`
- `list_attributes(2)`
- `query_objects(2)`
- `collection_result_display_limit(3)`
- `wildcards(3)`

get_buffers

Creates a collection of buffer cells.

SYNTAX

```
collection get_buffers
[-quiet]
[-regexp | -exact]
[-nocase]
[-filter expression]
[-inverter]
[-inverting_buffers]
[-library lib_spec]
[patterns]
```

Data Types

```
expression  string
lib_spec    collection
patterns    list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns_or_objects* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding "." to the beginning or end of the expressions, as needed.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each buffer cell in the collection, the expression is evaluated based on the buffer cell's attributes. If the expression evaluates to true, the buffer cell is included in the result.

To see the list of library cell attributes that you can use in the expression, use the **list_attributes -application -class lib_cell** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-inverter

Gets inverter cells only. By default, only buffer cells are returned. To include the library cells that have both inverting and noninverting outputs, use the **-inverting_buffers** option.

-inverting_buffers

Includes inverting buffer cells in addition to buffer cells (or inverter cells, if you use the **-inverter** option). Inverting buffers are library cells that have both inverting and noninverting outputs.

By default, only buffer cells are returned.

-library *lib_spec*

Specifies the libraries from which to collect the buffer cells. By default, the buffer cells are collected from all libraries in the link path.

patterns

Creates a collection of buffer cells whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page.

If you do not specify this option, the command uses the asterisk (*) as the default pattern.

DESCRIPTION

This command creates a collection of buffer cells that match the specified criteria.

The command returns a collection if any library cells match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries all buffer cells in the current library.

```
prompt> get_buffers  
{slow/BUF2X2 slow/BUF2X4 slow/BUF2X12}
```

The following example queries all inverter cells in the misc_cmos library.

```
prompt> get_buffers -inverter -library misc_cmos  
{misc_cmos/INV2X2 misc_cmos/INV2X4 misc_cmos/INVX12}
```

SEE ALSO

- collections(2)
- filter_collection(2)
- get_lib_cells(2)
- list_attributes(2)
- query_objects(2)
- collection_result_display_limit(3)
- wildcards(3)

get_cells

Creates a collection of cells from the current design, relative to the current instance.

SYNTAX

```
collection get_cells
  [-quiet]
  [-regexp | -exact]
  [-nocase]
  [-filter expression]
  [-hierarchical]
  [-rtl]
  [patterns]
  | -of_objects objects
  | -within region
  | -touching region
  | -intersect region
  | -at point
  [-all]
```

Data Types

```
expression  string
patterns    list
objects     collection
region      list
point       list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the

beginning or end of the expressions, as needed.

The **-regexp** and **-exact** options are mutually exclusive. You can specify only one of these options.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression.

For each cell in the collection, the expression is evaluated based on the cell's attributes. If the expression evaluates to true, the cell is included in the result.

To see the list of cell attributes that you can use in the expression, use the **list_attributes -application -class cell** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-hierarchical

Searches for cells level-by-level, relative to the current instance. The name of the object at a particular level must match the patterns. For example, if there is a cell block1/adder, a hierarchical search finds it by using "adder".

By default, the search is not hierarchical.

-rtl

Creates a collection of cells with the **object_rtl_name** attribute.

patterns

Creates a collection of cells whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

The *patterns*, **-of_objects**, **-object_id**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

-of_objects *objects*

Creates a collection of cells connected to the specified objects. Each object can be a named pin or net, a collection of pins, a collection of nets, or a power domain. When the object is a power domain, the cells returned are the root cells of the power domain.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses the * (asterisk) as the default pattern.

In addition, you cannot use the **-hierarchical** option with the **-of_objects** option.

-within *region*

Creates a collection that contains all cells that are completely inside the specified region and do not overlap the boundary. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} or {{*llx lly* *urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input

polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-object_id**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

In addition, you cannot use the **-hierarchical** option with the **-within** option.

-touching region

Creates a collection that contains all cells that are inside the specified region, including those that overlap the boundary. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is $\{\{lx\ ly\} \{urx\ ury\}\}$ or $\{lx\ ly\ urx\ ury\}$, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is $\{\{x1\ y1\} \{x2\ y2\} \dots \{xN\ yN\} \{x1\ y1\}\}$, where each $\{x\ y\}$ pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-object_id**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

In addition, you cannot use the **-hierarchical** option with the **-touching** option.

-intersect region

Creates a collection that contains all cells that intersect the boundary of the specified region and at least part of the cell is outside of the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is $\{\{lx\ ly\} \{urx\ ury\}\}$ or $\{lx\ ly\ urx\ ury\}$, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is $\{\{x1\ y1\} \{x2\ y2\} \dots \{xN\ yN\} \{x1\ y1\}\}$, where each $\{x\ y\}$ pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-object_id**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

In addition, you cannot use the **-hierarchical** option with the **-intersect** option.

-at point

Creates a collection that contains all cells at the specified point. The format for specifying a point is $\{x\ y\}$.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-object_id**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

In addition, you cannot use the **-hierarchical** option with the **-at** option.

-all

Includes physical-only cells in the collection. The following types of cells are considered physical-only cells:

- Standard cell fillers
- Pad fillers

- Corner cells
- Flip-chip pads (bumps)
- Chips
- Cover cells
- Tap cells
- Cells that have only power and ground ports

By default, physical-only cells are not included in the collection.

DESCRIPTION

This command creates a collection of cells that match the specified criteria. By default, the command creates the collection of cells from the current design, relative to the current instance.

If the command cannot find any cells that match the criteria and the current design is not linked, the design automatically links.

The command returns a collection if any cells match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the cells that begin with "o" and reference an FD2 library cell. Although the output looks like a list, it is only a display.

```
prompt> get_cells "o*" -filter "@ref_name == FD2"
{o_reg1 o_reg2 o_reg3 o_reg4}
```

The following example queries the cells connected to a collection of pins.

```
prompt> set pinsel [get_pins o*/CP]
{o_reg1/CP o_reg2/CP}
```

```
prompt> get_cells -of_objects $pinsel
{o_reg1 o_reg2}
```

The following example queries the cells connected to a collection of nets.

```
prompt> set netsel [get_nets tmp]
{tmp}
```

```
prompt> get_cells -of_objects $netset  
{b c}
```

The following example removes the wire load model from the i1 and i2 cells.

```
prompt> remove_wire_load_model [get_cells {i1 i2}]  
Removing wire load model from cell 'i1'.  
Removing wire load model from cell 'i2'.  
1
```

SEE ALSO

- `collections(2)`
- `filter_collection(2)`
- `list_attributes(2)`
- `query_objects(2)`
- `win_select_objects(2)`
- `collection_result_display_limit(3)`
- `wildcards(3)`

get_clocks

Creates a collection of clocks from the current design.

SYNTAX

```
collection get_clocks
  [-quiet]
  [-regex]
  [-nocase]
  [-filter expression]
  [patterns]
```

Data Types

```
expression  string
patterns    list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regex

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the == and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regex** command. When using the **-regex** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each clock in the collection, the expression is evaluated based on the clock's attributes. If the expression evaluates to true, the clock is included in the result.

To see the list of clock attributes that you can use in the expression, use the **list_attributes -application -class clock** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

patterns

Creates a collection of clocks whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

If you do not specify this option, the command uses the asterisk (*) as the default pattern.

DESCRIPTION

This command creates a collection of clocks that match the specified criteria.

The command returns a collection if any clocks match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example applies the **set_max_time_borrow** command to all clocks in the design that match the PHI* pattern.

```
prompt> set_max_time_borrow 0 [get_clocks "PHI*"]
```

The following example sets the clock_list variable to the collection of clocks that have a period of 15:

```
prompt> set clock_list [get_clocks * -filter "@period==15"]
```

SEE ALSO

collections(2)
filter_collection(2)
create_clock(2)
list_attributes(2)
query_objects(2)
report_clock(2)

collection_result_display_limit(3)
wildcards(3)

get_clusters

Creates a collection of one or more clusters.

SYNTAX

```
collection get_clusters
  [-quiet]
  [-regex [-nocase]]
  [-exact]
  [-filter expression]
  [-hier]
  [-flat]
  patterns | -of_objects objects
```

Data Types

```
expression  string
patterns    list
objects     list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regex

Views the *patterns* option as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The **-regex** and **-exact** options are mutually exclusive; you can choose only one.

-nocase

Makes matches case-insensitive when combined with **-regex**. You can use the **-nocase** option only when you also use **-regex**.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * (asterisk) and ? (question mark) wildcard characters. The **-regex** and **-exact** options are mutually exclusive; you can choose only one.

-filter *expression*

Filters the collection with the value of *expression*. For any clusters that match the *patterns* value, the expression is evaluated based on the attributes of the cluster. If the expression evaluates to *true*, the design is included in the result.

-hier

Specifies hierarchical.

-flat

Specifies flat.

patterns

Matches cluster names against patterns. Patterns can include the wildcard character * (asterisk) and ? (question mark), or regular expressions based on the **-regexp** option.

-of_objects *objects*

Get cells related to these objects.

DESCRIPTION

This command creates a collection of clusters that match certain criteria. It returns a collection handle (identifier) if any clusters match the *patterns* or *objects* and pass the filter (if specified). If no objects match your criteria, the command returns an empty string.

You can use the **get_clusters** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_clusters** result to a variable.

When issued from the command prompt, the **get_clusters** command behaves as though the **query_objects** command has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change the maximum using the **collection_result_display_limit** variable.

The "implicit query" property of **get_clusters** provides a fast, simple way to display clusters in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_clusters** as an argument to the **query_objects** command. Add the **-query** option to emulate the behavior of **query_objects**.

For information about collections and querying objects, see the **collections** man page.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example queries the clusters that begin without. *Although the output looks like a list, it is only a display.*

```
prompt> get_clusters out*
{out_reg}
```

SEE ALSO

[collections\(2\)](#)

```
filter_collection(2)
query_objects(2)
regexp(2)
collection_result_display_limit(3)
```

get_command_option_values

Queries current or default option values.

SYNTAX

```
get_command_option_values  
[-default | -current]  
-command command_name
```

Data Types

command_name string

ARGUMENTS

-default

Gets the default option values, if available.

-current

Gets the current option values, if available.

-command *command_name*

Gets the option values for this command.

DESCRIPTION

This command attempts to query a default or current value for each option (of the command) that has default and/or current-value-tracking enabled. Details of how the option value is queried depend on whether one of the **-current** or **-default** options is specified (see below).

A "Tcl array set compatible" (possibly empty) list of option names and values is returned as the Tcl result. The even-numbered entries in the list are the names of options that were enabled for default-value-tracking or current-value-tracking and had at least one of these values set to a not-undefined value). Each odd-numbered entry in the list is the default or current value of the option name preceding it in the list.

Any options that were not enabled for either default-value-tracking nor current-value-tracking are omitted from the output list. Similarly, options that were enabled for default-value-tracking or current-value-tracking, but for which no (not-undefined) default or current value is set, are omitted from the result list.

If neither **-current** nor **-default** is specified, then for each command option that has either default-value-tracking or current-value-tracking (or both) enabled, the value returned is as follows:

- The current value is returned if current-value-tracking is enabled and a (not-undefined) current value has been set;
- Otherwise the default value is returned if default-value-tracking is enabled and a (not-undefined) default value has been set;
- Otherwise the name and value pair for the option is not included in the result list.

If **-current** is specified, the value returned for an option is the current value if current-value-tracking is enabled, and a (not-undefined) current value has been set; otherwise the name and value pair for the option is omitted from the result list.

If **-default** is specified, the value returned for an option is the default value if default-value-tracking is enabled, and a (not-undefined) default value has been set; otherwise the name and value pair for the option is omitted from the result list.

The result list from **get_command_option_values** includes option values of both dash options and positional options (assuming that both kinds of options of a command have been enabled for value-tracking).

The command issues a Tcl error in a variety of situations, such as if an invalid command name was passed in with **-command**.

EXAMPLES

The following example shows the use of **get_command_option_values**:

```
prompt> test -opt1 10 -opt2 20
1
```

```
prompt> get_command_option_values -command test
-bar1 10 -bar2 20
```

SEE ALSO

get_cross_probing_info

Gets cross-probing information about cells and ports and returns the results as a string.

SYNTAX

```
string get_cross_probing_info  
    [-unique_source]  
    objects
```

Data Types

objects collection

ARGUMENTS

-unique_source

Prevents duplicate source file and line number combinations from being returned in the result.

If you specify this option, the string that is returned follows a different format; it does not use a pair of braces between the different source positions of an object or the results between different objects. Instead, the tool returns the results in the following format:

obj1_file1:obj1_line1 obj2_file1:obj2_line1 obj2_file2:obj2_line2

If an object has no known source position or valid file and line number, nothing is returned for the object.

objects

Specifies the cell and port objects to query.

DESCRIPTION

This command provides cross-probing information about specified cells and ports and returns the results as a string.

You must have a design that was analyzed and elaborated or synthesized using Design Compiler version I-2013.12 or later. The tool provides information about where the cell or port was created, specifying which file and line number the object came from. Some objects in the design might have been merged, which can result in an object with multiple source locations. In this case, multiple results are returned for the object.

The results of each object are separated by a pair of braces between each source position and between each object. For example, if you specify two objects with the **get_cross_probing_info** command, obj1 and obj2, where obj1 has one source position and obj2 has two source positions, the tool returns the results in the following format:

```
{ { obj1_file1:obj1_line1 } } { { obj2_file1:obj2_line1 } {obj2_file2:obj2_line2 } }
```

You can prevent duplicate file and line number combinations in the results by using the **-unique_source** option. In this case, the tool returns the results in the following format:

```
obj1_file1:obj1_line1 obj2_file1:obj2_line1 obj2_file2:obj2_line2
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows results from the **get_cross_probing_info** command:

```
prompt> get_cross_probing_info [get_cells -hierarchical]
{ {/usr/joe/design/m.v:21} } { {/usr/joe/design/m.v:22} } { {/usr/joe/design/m.v:21} }
{ {/usr/joe/design/m.v:22} } { {/usr/joe/design/m.v:28} } { {/usr/joe/design/m.v:12} }
{ {/usr/joe/design/m.v:13} } { {/usr/joe/design/m.v:14} } { {/usr/joe/design/m.v:4} }
```

The following example shows results from the **get_cross_probing_info** command with the **-unique_source** option:

```
prompt> get_cross_probing_info [get_cells -hierarchical] -unique_source
/usr/joe/design/m.v:21 /usr/joe/design/m.v:22 /usr/joe/design/m.v:28 /usr/joe/design/m.v:12
/usr/joe/design/m.v:13 /usr/joe/design/m.v:14 /usr/joe/design/m.v:4
```

SEE ALSO

report_cross_probing(2)
cross_probing_filter(2)

get_defined_commands

Get information on defined commands and groups.

SYNTAX

```
string get_defined_commands  
    [-details]  
    [-groups]  
    [pattern]
```

Data Types

pattern string

ARGUMENTS

-details

Get detailed information on specific command or group.

-groups

Search groups rather than commands

pattern

Return commands or groups matching pattern. The default value of this argument is "".

DESCRIPTION

The **get_defined_commands** gets information about defined commands and command groups. By default the command returns a list of commands that match the specified pattern.

When **-details** is specified, the return value is a list that is suitable as input to the **array set** command. The returned value is a list with an even number of arguments. Each odd-numbered element in the list is a key name, and each even-numbered element in the list is the value of the previous key. The **-details** option is only legal if the pattern matches exactly one command or group.

When **-group** is specified with **-details**, the supported keys are as follows:

name

This key contains the name of the group.

info

This key contains the short help for the group.

commands

This key contains the commands in the group

When **-details** is used with a command, the supported keys are as follows:

name

This key contains the name of the command.

info

This key contains the short help for the command.

groups

This key contains the group names that this command belongs to.

options

This key contains the options defined for the command. The value is a list.

return

This key contains the return type for the command.

Each element in the *options* list also follows the key value pattern. The set of available keys for options are as follows:

name

This key contains the name of the option.

info

This key contains the short help for the option.

value_info

This key contains the short help string for the value

type

The type of the option.

required

The value will be 0 or 1 depending if the option is optional or required.

is_list

Will be 1 if the option requires a list.

list_length

If a list contains the list length constraint. One of any, even, odd, non_empty or a number of elements.

allowed_values

The allowed values if the option has specified allowed values.

min_value

The minimum allowed value if the option has specified one.

max_value

The maximum allowed value if the option has specified one.

EXAMPLES

```
prompt> get_defined_commands *collection
add_to_collection append_to_collection copy_collection filter_collection
foreach_in_collection index_collection sort_collection
prompt> get_defined_commands -details sort_collection
name sort_collection info {Create a sorted copy of the collection}
groups {} options {{name -descending info {Sort in descending order}
value_info {} type boolean required 1 is_list 0} {name -dictionary info
{Sort strings dictionary order.} value_info {} type boolean required 1
is_list 0} {name collection info {Collection to sort} value_info
collection type string required 0 is_list 0} {name criteria info {Sort
criteria - list of attributes} value_info criteria type list required 0
is_list 1 list_length non_empty}}
```

SEE ALSO

help(2)
man(2)

get_design_lib_path

Returns the directory to which the specified library is mapped.

SYNTAX

```
status get_design_lib_path  
      library_name
```

Data Types

library_name string

ARGUMENTS

library_name

Specifies the library whose directory mapping is returned.

DESCRIPTION

This command returns the directory to which the specified library is mapped.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example maps the library named *MY_LIB* to the same directory as the *WORK* library:

```
prompt> define_design_lib MY_LIB \  
      -path get_design_lib_path("WORK")
```

SEE ALSO

analyze(2)
define_design_lib(2)
elaborate(2)
read_file(2)
report_design_lib(2)

get_designs

Creates a collection of one or more designs loaded into the tool.

SYNTAX

```
collection get_designs
  [-hierarchical]
  [-quiet]
  [-regex]
  [-nocase]
  [-exact]
  [-filter expression]
  patterns
```

Data Types

```
expression  string
patterns    list
```

ARGUMENTS

-hierarchical

Searches for designs inferred by the design hierarchy relative to the current instance. The full name of the object at a particular level must match the patterns. The use of this option does not force an auto link.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regex

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * (asterisk) and ? (question mark) wildcard characters.

-filter *expression*

Filters the collection with *expression*. For any designs that match *patterns*, the expression is evaluated based on the design's

attributes. If the expression evaluates to true, the design is included in the result.

patterns

Matches design names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards refer to the **wildcards** man page.

DESCRIPTION

The **get_designs** command creates a collection of designs from those currently loaded into the tool that match certain criteria. The command returns a collection if any designs match the *patterns* and pass the filter (if specified). If no objects match your criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored. The expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_designs** command at the command prompt, or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the **get_designs** result to a variable.

When issued from the command prompt, **get_designs** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum using the **collection_result_display_limit** variable.

The implicit query property of **get_designs** provides a fast, simple way to display designs in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_designs** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the designs that begin with *mpu*. Although the output looks like a list, it is just a display. A complete listing of designs is available using the **list_designs** command.

```
prompt> get_designs mpu*
{mpu_0_0 mpu_0_1 mpu_1_0 mpu_1_1}
```

The following example shows that, given a collection of designs, you can remove those designs:

```
prompt> remove_design [get_designs mpu*]
Removing design mpu_0_0...
Removing design mpu_0_1...
Removing design mpu_1_0...
Removing design mpu_1_1...
```

SEE ALSO

`collections(2)`
`filter_collection(2)`
`list_designs(2)`
`query_objects(2)`
`remove_design(2)`
`collection_result_display_limit(3)`
`wildcards(3)`

get_dft_hierarchical_pins

Returns a collection of ports and hierarchical pins created during DFT insertion.

SYNTAX

collection **get_dft_hierarchical_pins**

ARGUMENTS

The **get_dft_hierarchical_pins** command has no arguments.

DESCRIPTION

This command returns a collection of ports and hierarchical pins created during DFT insertion command in the current design.

EXAMPLES

The following example shows how to use the **get_dft_hierarchical_pins** command to report attributes for each port created during DFT insertion:

```
prompt> set ports [get_ports -quiet [get_dft_hierarchical_pins]]
{data_source test_mode test_si1 test_si2 test_so2 test_se}
```

```
prompt> foreach_in_collection p $ports {report_attribute $p}
```

```
*****
```

```
Report : Attribute
```

```
Design : cntl
```

```
Version: N-2017.09-SP4
```

```
Date   : Wed Apr 11 11:01:04 2018
```

```
*****
```

Design	Object	Type	Attribute Name	Value
cntl	data_source	port	created_during_dft	true
cntl	data_source	port	created_by_test_compiler	true

```
cntl      data_source      port      is_test_circuitry      true
```

```
*****
```

```
Report : Attribute
```

```
Design : cntl
```

```
Version: N-2017.09-SP4
```

```
Date   : Wed Apr 11 11:01:04 2018
```

```
*****
```

Design	Object	Type	Attribute Name	Value
cntl	test_mode	port	created_during_dft	true
cntl	test_mode	port	created_by_test_compiler	true
cntl	test_mode	port	is_test_circuitry	true

```
*****
```

```
Report : Attribute
```

```
Design : cntl
```

```
Version: N-2017.09-SP4
```

```
Date   : Wed Apr 11 11:01:04 2018
```

```
*****
```

Design	Object	Type	Attribute Name	Value
cntl	test_si1	port	created_during_dft	true
cntl	test_si1	port	created_by_test_compiler	true
cntl	test_si1	port	is_test_circuitry	true

```
*****
```

```
Report : Attribute
```

```
Design : cntl
```

```
Version: N-2017.09-SP4
```

```
Date   : Wed Apr 11 11:01:04 2018
```

```
*****
```

Design	Object	Type	Attribute Name	Value
cntl	test_si2	port	created_during_dft	true
cntl	test_si2	port	created_by_test_compiler	true
cntl	test_si2	port	is_test_circuitry	true

```
*****
```

```
Report : Attribute
```

```
Design : cntl
```

```
Version: N-2017.09-SP4
```

```
Date   : Wed Apr 11 11:01:05 2018
```

```
*****
```

Design	Object	Type	Attribute Name	Value
cntl	test_so2	port	created_during_dft	true
cntl	test_so2	port	created_by_test_compiler	true
cntl	test_so2	port	is_test_circuitry	true

```
*****
```

```
Report : Attribute
```

Design : cntl
Version: N-2017.09-SP4
Date : Wed Apr 11 11:01:05 2018

Design	Object	Type	Attribute Name	Value
cntl	test_se	port	created_during_dft	true
cntl	test_se	port	created_by_test_compiler	true
cntl	test_se	port	is_test_circuitry	true

SEE ALSO

report_dft_hierarchical_pins(2)
current_design(2)
insert_dft(2)

get_die_area

Returns a collection containing the die area of the current design.

SYNTAX

string **get_die_area**

ARGUMENTS

None

DESCRIPTION

This command returns a collection containing the die area of the current design. If the die area is not defined, the command returns an empty string. The die area is also known as the cell boundary. It represents the silicon boundary of a chip and typically encloses all objects of a design, such as pads, I/O pins, cells, and so on. There should be only one die area in a design. Use the **get_attribute** command to get information about the die area, such as the object class, bounding box information, coordinates of the die area for the current design, die area boundary, and die area name. You can use the **report_attribute** command to browse attributes.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

Use the **get_attribute** command to get information about the die area. The following example shows how to return the object class using the **object_class** attribute:

```
prompt> get_attribute [get_die_area] object_class
die_area
```

The following example shows how to return the bounding box information and the coordinates of the die area for the current design using the **bbox** attribute:

```
prompt> get_attribute [get_die_area] bbox
```

```
{0.000 0.000} {400.000 400.000}
```

The following example shows how to return the die area name using the **name** attribute:

```
prompt> get_attribute [get_die_area] name  
{my_design_die_area}
```

SEE ALSO

```
set_placement_area(2)
```

get_dont_touch_cells

Creates a collection of dont_touch cells that meet the specified criteria.

SYNTAX

```
collection get_dont_touch_cells  
  [-type type]  
  [-hierarchical]  
  patterns
```

Data Types

```
type      string  
patterns  collection
```

ARGUMENTS

-type *type*

Matches the cell with the specified dont_touch type. You can use a wildcard.

To list the dont_touch types for cells, use the **list_dont_touch_type** command with the **-class cell** option.

-hierarchical

Searches for cells in the hierarchy, relative to the current instance. The full name of the object at a particular level must match the patterns. For example, if the specified cell name is block1/cell2, a hierarchical search finds a cell named cell2 in a block named block1.

patterns

Matches the cell names with the specified patterns. A collection of cells is also accepted.

DESCRIPTION

This command creates a collection of dont_touch cells in the current design, relative to the current instance, that match the specified criteria. If no cells match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as the **query_objects** command. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report

the objects in the collection. By default, the command displays a maximum of 100 objects. You can change this value by setting the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

For information about the dont_touch types that can be specified for cells, use the **list_dont_touch_type -class cells** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the dont_touch cells with an **ideal_network** setting under the block named InstDecode:

```
prompt> get_dont_touch_cells -type idn InstDecode/*  
{InstDecode/reset_UPF_LS InstDecode/U38 InstDecode/U36}
```

The following examples queries the dont_touch cells with a DFT-related dont_touch setting under the block named core:

```
prompt> get_dont_touch_cells -type dft* core/*  
{core/U34 core/U35 core/U89}
```

SEE ALSO

- list_dont_touch_types(2)
- report_dont_touch(2)
- get_dont_touch_nets(2)
- set_dont_touch(2)
- set_dont_touch_network(2)
- collection_result_display_limit(3)

get_dont_touch_nets

Creates a collection of dont_touch nets that meet the specified criteria.

SYNTAX

```
collection get_dont_touch_nets  
  [-type type]  
  [-hierarchical]  
  patterns
```

Data Types

```
type      string  
patterns  collection
```

ARGUMENTS

-type *type*

Matches the net with the specified dont_touch type. You can use a wildcard.

To list the dont_touch types for nets, use the **list_dont_touch_type** command with the **-class net** option.

-hierarchical

Searches for nets in the hierarchy, relative to the current instance. The full name of the object at a particular level must match the patterns. For example, if the specified net name is block1/net2, a hierarchical search finds a net named net2 in block named block1.

patterns

Matches the net names with the specified patterns. A collection of nets is also accepted.

DESCRIPTION

This command creates a collection of dont_touch nets in the current design, relative to the current instance, that match the specified criteria. If no nets match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as the **query_objects** command. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** to report the objects

in the collection. By default, the command displays a maximum of 100 objects. You can change this value by setting the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

For information about the dont_touch types that can be specified for nets, use the **list_dont_touch_type -class nets** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the nets whose names begin with NET in a block named block1.

```
prompt> get_dont_touch_nets -type mv* block1/NET*  
{block1/NET1QNX block1/NET2QNX}
```

The following example queries the nets under the block named block1 that have a dont_touch type of dtn.

```
prompt> get_dont_touch_nets -type dtn block1/*  
{block1/clock}
```

SEE ALSO

- collections(2)
- get_dont_touch_cells(2)
- get_nets(2)
- list_dont_touch_types(2)
- report_dont_touch(2)
- set_dont_touch(2)
- set_dont_touch_network(2)
- collection_result_display_limit(3)
- wildcards(3)

get_flat_cells

Creates a collection of leaf cells that match certain criteria in the current design.

SYNTAX

```
collection get_flat_cells
  [-quiet]
  [-regexp | -exact]
  [-nocase]
  [-filter expression]
  [patterns | -of_objects objects]
  [-all]
```

Data Types

```
expression  string
patterns    list
objects     list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding "." to the beginning or end of the expressions, as needed.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each cell in the collection, the expression is evaluated based on the cell's attributes. If the expression evaluates to true, the cell is included in the result.

To see the list of cell attributes that you can use in the expression, use the **list_attributes -application -class cell** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

patterns

Creates a collection of cells whose full names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

When using the *patterns* argument, the command searches all leaf cells to match the *patterns* argument on their full names regardless of their hierarchical level.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses the * (asterisk) as the default pattern.

Creates a collection of cells connected to the specified objects. Each object must be a named cell, pin, net, or a collection of these objects.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses the * (asterisk) as the default pattern.

-all

Includes physical-only cells in the collection. The following types of cells are considered physical-only cells:

- Standard cell fillers
- Pad fillers
- Corner cells
- Flip-chip pads (bumps)
- Chips
- Cover cells
- Tap cells
- Cells that have only power and ground ports

By default, physical-only cells are not included in the collection.

DESCRIPTION

This command creates a collection of leaf cells from the current design that match the specified criteria. Leaf cells are those processed during place and route. They are shown and can be selected in the GUI layout window.

The command returns a collection if any leaf cells match the criteria. If no objects match the criteria, the command returns an empty string.

The **get_flat_cells** and **get_cells** commands both return cell objects. The collections returned by both commands are interchangeable and can be passed along to any command that accepts cell objects.

However, the **get_flat_cells** command differs from the **get_cells** command in several ways. The **get_cells** command performs the pattern search based on the logical hierarchy of the design and is consistent with the commands used for synthesis and timing analysis that have been in Design Compiler, Primetime, and SDC. The **get_flat_cells** command searches all leaf cells to match the *patterns* argument on their full names relative to the current design, it does not depend on the current instance. It provides pattern search in a way that is consistent with the place-and-route view of the design, and thus is simpler for many tasks in the IC Compiler flow. You can use whichever command is easier depending on the type of search you are doing.

Note that the **get_flat_cells** command is not part of the SDC format standard and should not be used in timing constraint scripts if you want to apply them across other tools or by using the **read_sdc** command. Also, in a design with multiple physical hierarchies, the **get_flat_cells** command does not search cells in child blocks.

Due to the pattern matching behavior of this command, the search space is often larger than the corresponding **get_cells** command. As a result, the runtime might be much larger and you should limit the number of calls to this command that contain regular expressions or wildcards.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects** or **change_selection**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries all black box cells in the current design. Although the output looks like a list, it is only a display.

```
prompt> get_flat_cells * -filter {is_black_box == true }
{cornerur cornerul cornerlr cornerll I_CLOCK_GEN/I_PLL_SD I_CLOCK_GEN/I_PLL_PCI
I_CLOCK_GEN/I_CLKMUL I_ORCA_TOP/I_BLENDER/latched_clk_en_reg I_ORCA_TOP/I_
CONTEXT_MEM/CONTEXT_RAM_3 I_ORCA_TOP/I_CONTEXT_MEM/CONTEXT_RAM_2 ...}
```

The following example queries the leaf cells whose full name begins with "BLOCK/", as well as all leaf cells any number of logical levels below the BLOCK hierarchical cell.

```
prompt> get_flat_cells BLOCK/*
{BLOCK/SB/U1 BLOCK/SB/U2 BLOCK/SB/U3 BLOCK/SB/U4 BLOCK/U5 BLOCK/U6
BLOCK/U7 BLOCK/U8 BLOCK/U9}
```

As a comparison to the previous example, the following example finds only cells that are direct children of the BLOCK hierarchical cell.

```
prompt> get_cells BLOCK/*
{BLOCK/SB BLOCK/U9 BLOCK/U5 BLOCK/U8 BLOCK/U7 BLOCK/U6}
```

The following example shows that, given a collection of nets, you can select the leaf cells connected to those nets. You can see that these cells are selected in the GUI layout view if there is one open for the current design.

```
prompt> set netsel [get_flat_nets r1_2_]
{r1_2_}
prompt> change_selection [get_flat_cells -of_object $netsel]
```

SEE ALSO

collections(2)
filter_collection(2)
get_cells(2)
get_flat_pins(2)
get_flat_nets(2)
list_attributes(2)
query_objects(2)
change_selection(2)
collection_result_display_limit(3)
wildcards(3)

get_flat_nets

Creates a collection of top-level nets of hierarchical net groups in the current design that match the specified criteria.

SYNTAX

```
collection get_flat_nets
  [-quiet]
  [-regexp | -exact]
  [-nocase]
  [-filter expression]
  [patterns | -of_objects objects]
  [-all]
```

Data Types

```
expression  string
patterns    list
objects     list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding "." to the beginning or end of the expressions, as needed.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each net in the collection, the expression is evaluated based on the net's attributes. If the expression evaluates to true, the net is included in the result.

To see the list of net attributes that you can use in the expression, use the **list_attributes -application -class net** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

patterns

Creates a collection of top-level nets of hierarchical net groups whose full names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

When using the *patterns* argument, the command searches all top-level nets of hierarchical net groups to match the *patterns* argument on their full name.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses the asterisk (*) as the default pattern.

Creates a collection of top-level nets of hierarchical net groups that are connected to the specified objects. Each object is either a named pin, port, net, cell, or a collection of these objects.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses the asterisk (*) as the default pattern.

-all

Includes power and ground nets.

DESCRIPTION

This command creates a collection of top-level nets of hierarchical net groups in the current design that match the specified criteria.

The **get_flat_nets** and **get_nets** commands return both return net objects. The collections returned by both commands are interchangeable and can be passed along to any command that accept net objects.

However, the **get_flat_nets** command differs from the **get_nets** command in several ways. The **get_nets** command performs the pattern search based on the logical hierarchy of the design and is consistent with the commands used for synthesis and timing analysis that have been in Design Compiler, PrimeTime, and SDC. The **get_flat_nets** command searches the top-level nets of hierarchical net groups to match the *patterns* argument on their full names relative to the current design, it does not depend on the current instance. It provides pattern search in a way that is consistent with the place-and-route view of the design, and thus is simpler for many tasks in the IC Compiler flow. You can use whichever command is easier depending on the type of search you are doing.

Note that the **get_flat_nets** command is not part of the SDC format standard and should not be used in timing constraint scripts if you want to apply them across other tools or by using the **read_sdc** command. Also in a design with multiple physical hierarchies, the **get_flat_nets** command does not search nets in child blocks.

Due to the pattern matching behavior of this command, the search space is often larger than the corresponding **get_nets** command. As a result, the runtime might be much larger and you should limit the number of calls to this command that contain regular expressions or wildcards.

The command returns a collection if any nets match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects** or **change_selection**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the top-level nets whose full name begins with "BLOCK/". Although the output looks like a list, it is just a display.

```
prompt> get_flat_nets BLOCK/*
{BLOCK/SB/n76}
```

The following example shows that there is no top-level net whose full name begins with "BLOCK/" and ends with "n1".

```
prompt> get_flat_nets BLOCK/*n1
Warning: No nets matched 'BLOCK/*n1' (SEL-004)
```

As a comparison to the previous example, the following example shows that there is a net whose full name begins with "BLOCK/" and ends with "n1".

```
prompt> get_nets BLOCK/*n1
{BLOCK/n1}
```

The following selects the top-level nets of hierarchical net groups that connect to cells. You can see that these nets are selected in the GUI layout view if there is one open for the current design.

```
prompt> set cellsel [get_flat_cells I_ORCA_TOP/I_SDRAM_IF/U23276]
{I_ORCA_TOP/I_SDRAM_IF/U23276}
```

```
prompt> change_selection [get_flat_nets -of_objects $cellsel]
```

SEE ALSO

collection_result_display_limit(3)
collections(2)
filter_collection(2)
get_flat_cells(2)
get_flat_pins(2)
get_nets(2)
list_attributes(2)

query_objects(2)
wildcards(3)

get_flat_pins

Creates a collection of leaf-cell pins in the current design that match the specified criteria.

SYNTAX

```
collection get_flat_pins
  [-quiet]
  [-regexp | -exact]
  [-nocase]
  [-filter expression]
  [patterns | -of_objects objects]
  [-all]
```

Data Types

```
expression  string
patterns    list
objects     list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding "." to the beginning or end of the expressions, as needed.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each pin in the collection, the expression is evaluated based on the pin's attributes. If the expression evaluates to true, the pin is included in the result.

To see the list of pin attributes that you can use in the expression, use the **list_attributes -application -class pin** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

patterns

Creates a collection of leaf-cell pins whose full names match the specified patterns. Patterns can include the asterisk (*) and question mark (?) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

When using the *patterns* argument, the command searches all pins of leaf cells to match the *patterns* argument on their full names regardless of their hierarchical level.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses the * (asterisk) as the default pattern.

-of_objects *objects*

Creates a collection of leaf-cell pins connected to the specified objects. Each object is a named leaf cell, a net, or a collection of these objects.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses the asterisk (*) as the default pattern.

-all

This option has no effect. Added for script compatibility.

DESCRIPTION

This command creates a collection of leaf-cell pins in the current design that match the specified criteria. Leaf cells are those processed during place and route. They are shown and can be selected in GUI layout window.

The **get_flat_pins** and **get_pins** commands both return pin objects. The collections returned by both commands are interchangeable and can be passed along to any command that accepts pin objects.

However, the **get_flat_pins** command differs from the **get_pins** command in several ways. The **get_pins** command performs the pattern search based on the logical hierarchy of the design and is consistent with the commands used for synthesis and timing analysis that have been in Design Compiler, PrimeTime, and SDC. The **get_flat_pins** command searches all pins of leaf cells to match the *patterns* argument on their full names relative to the current design; it does not depend on the current instance. It provides pattern search in a way that is consistent with the place-and-route view of the design, and thus is simpler for many tasks in the IC Compiler flow. You can use whichever command is easier depending on the type of search you are doing.

Note that the **get_flat_pins** command is not part of the SDC format standard and should not be used in timing constraint scripts if you want to apply them across other tools or by using the **read_sdc** command. Also, in a design with multiple physical hierarchies, the **get_flat_pins** command does not search pins in child blocks.

Due to the pattern matching behavior of this command, the search space is often larger than the corresponding **get_pins** command.

As a result, the runtime might be much larger and you should limit the number of calls to this command that contain regular expressions or wildcards.

The command returns a collection if any pins match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects** or **change_selection**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the I pins of leaf cells that begin with "I_CLK". Although the output looks like a list, it is just a display.

```
prompt> get_flat_pins I_CLK*/I
{I_CLK_SOURCE_SDRAM_CLK/I I_CLK_SOURCE_SYS_2x_CLK/I
I_CLK_SOURCE_SYS_CLK/I I_CLK_SOURCE_PCLK/I}
```

The following example queries the ZN pins of all leaf cells any number of logical levels below the BLOCK hierarchical cell.

```
prompt> get_flat_pins BLOCK/*ZN
{BLOCK/SB/U1/ZN BLOCK/SB/U2/ZN BLOCK/SB/U3/ZN BLOCK/SB/U4/ZN
BLOCK/U5/ZN BLOCK/U6/ZN BLOCK/U7/ZN BLOCK/U8/ZN BLOCK/U9/ZN}
```

As a comparison to the previous example, the following example finds the ZN pins of the cells that are direct children of the BLOCK hierarchical cell.

```
prompt> get_pins BLOCK/*ZN
{BLOCK/U9/ZN BLOCK/U5/ZN BLOCK/U8/ZN BLOCK/U7/ZN BLOCK/U6/ZN}
```

The following example selects pins of leaf cells whose name begins with "O_1". You can see that these pins are selected in the GUI layout view if there is one open for the current design.

```
prompt> set csel [get_flat_cells O_1*]
{O_10__reg O_11__reg O_12__reg O_13__reg O_1__reg}
prompt> change_selection [get_flat_pins -of_objects $csel]
```

SEE ALSO

collections(2)
 filter_collection(2)
 get_pins(2)
 get_flat_cells(2)

```
get_flat_nets(2)
list_attributes(2)
query_objects(2)
collection_result_display_limit(3)
wildcards(3)
```

get_generated_clocks

Creates a collection of generated clocks.

SYNTAX

```
collection get_generated_clocks
  [-quiet]
  [-regexp | -exact]
  [-nocase]
  [-filter expression]
  [patterns]
```

Data Types

```
expression  string
patterns    list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each generated clock in the collection, the expression is evaluated based on the generated clock's attributes. If the expression evaluates to true, the generated clock is included in the result.

To see the list of clock attributes that you can use in the expression, use the **list_attributes -application -class clock** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

patterns

Creates a collection of generated clocks whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

If you do not specify this argument, the command uses the * (asterisk) as the default pattern.

DESCRIPTION

This command creates a collection of generated clocks in the current design that match the specified criteria.

To create a generated clock in the design, use the **create_generated_clock** command. To remove a generated clock from the design, use the **remove_generated_clock** command. To show information about clocks and generated clocks in the design, use the **report_clock** command.

The command returns a collection if any generated clocks match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command applies the **set_clock_latency** command on all generated clocks in the design whose names start with "GEN".

```
prompt> set_clock_latency 2.0 [get_generated_clocks "GEN*"]
```

The following command removes all the generated clocks in the design whose names start with "GEN1".

```
prompt> remove_generated_clock [get_generated_clocks "GEN1*"]
```

SEE ALSO

collections(2)
create_generated_clock(2)
filter_collection(2)
list_attributes(2)
query_objects(2)
remove_generated_clock(2)
report_clock(2)
collection_result_display_limit(3)

get_gui_stroke_bindings

Queries the data for stroke bindings.

SYNTAX

```
get_gui_stroke_bindings  
[-dictionary dict_name]  
[-builtin]
```

ARGUMENTS

-dictionary *dictionary_name*

Report only the bindings for the specified dictionary.

-builtin

Return information on the built-in commands that are available for use in bindings.

DESCRIPTION

This command queries the current state of stroke bindings for the application. If no options are specified, the application returns data for all dictionaries. The **-dictionary** option limits the data returned to only that for a given dictionary. The **-builtin** option queries for the built-in commands that are available for use with bindings.

A user does not typically use this command. The tool provides built-in reporting commands that produce human-readable reports from the information returned by this command. You can use the **report_gui_stroke_bindings** and **report_gui_stroke_builtins** commands for report generation.

This command returns its data as a Tcl list, suitable for further processing into reports or command streams by Tcl procedures.

DATA FORMAT

The command returns data differently depending on the options that are provided.

The data for a dictionary is a list of entries where each entry contains the stroke sequence and the data for the command invoked by that sequence. If the command queries more than one dictionary, another level of lists is added with dictionary name followed by dictionary data.

Built-in command data is formatted as a list of commands. Each command has the name of the built-in command followed by the data for the command.

EXAMPLES

The following example returns all binding information for all dictionaries.

```
psyn_gui-t> get_gui_stroke_bindings
```

The following example returns the bindings for the Graphics dictionary.

```
psyn_gui-t> get_gui_stroke_bindings -dictionary Graphics
```

The following example returns the built-in commands available for use in bindings.

```
psyn_gui-t> get_gui_stroke_bindings -builtin
```

SEE ALSO

```
report_gui_stroke_bindings(2)  
report_gui_stroke_builtins(2)  
set_gui_stroke_binding(2)  
set_gui_stroke_preferences(2)
```

get_latch_loop_groups

Returns a list of collections of pins, each collection containing the transparent latch data pins in one latch loop group.

SYNTAX

```
list get_latch_loop_groups  
  [-of_objects pin_list]  
  [-loop_breakers_only]
```

Data Types

pin_list list

ARGUMENTS

-of_objects *pin_list*

A collection of data pins of transparent latches. Only the pins of loop groups containing the specified pins are returned. By default, the command returns one collection for each loop group in the design.

-loop_breakers_only

Restricts the returned collection to containing only loop-breaker latch data pins. These are the pins that have their **is_latch_loop_breaker** attribute set to **true**. By default, all transparent latch data pins within the loop group are returned in the collections.

DESCRIPTION

When sequential loops of transparent latches exist in a design, a loop group containing all the latch data pins of intersecting loops is formed. The command **get_latch_loop_groups** analyzes the design and returns a collection for each loop group formed. The collection contains the latch data pins of the loop group. Combinational pins are not included in the results.

You can use the results of the **get_latch_loop_groups** command to find paths within loops that violate timing. See the example below.

This command has no effect if the variable **timing_enable_through_paths** is set to **false**.

EXAMPLES

The following example uses **get_latch_loop_groups** to report violating paths within latch loops. No paths outside the loops are reported.

```
prompt> foreach a_loop_group [get_latch_loop_groups -loop_breakers_only] {  
  report_timing -from $a_loop_group -to $a_loop_group -slack_lesser_than 0.0  
}
```

SEE ALSO

set_latch_loop_breakers(2)
report_latch_loop_groups(2)
timing_enable_through_paths(3)

get_lib_attribute

Returns the value of an attribute on a list of library objects.

SYNTAX

```
list get_lib_attribute  
    object_list  
    attribute_name
```

Data Types

```
object_list    list  
attribute_name string
```

ARGUMENTS

object_list

Lists the library objects for which the attribute value is to be returned.

attribute_name

Specifies the name of the attribute whose value is to be returned.

DESCRIPTION

This command searches *object_list* for the specified attribute and returns a list of attribute values. If the attribute is not found on any of the specified objects, the command returns an empty list.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to find the cell named */INV* and obtain its area value:

```
prompt> get_lib_attribute sample/INV area
```

```
Performing get_lib_attribute on library object  
'sample/INV'.  
{1.0}
```

The following example shows an error issued because the command does not find a cell named *INV1*:

```
prompt> get_attribute sample/INV1 area  
Error: The 'INV1' object does not exist in the 'sample' technology library.  
(UIL-54)  
{}
```

The following example shows the **lib_udg_attr** value returned for the user-defined group named *lib_udg1* in the library named *sample*:

```
prompt> get_lib_attribute sample/lib_udg1 lib_udg_attr  
Performing get_lib_attribute on library object 'sample/lib_udg1'.  
{Test}
```

The following example shows the area values returned for the gates named *G1* and *G2* in the library named *tech_lib*:

```
prompt> get_lib_attribute {sample/G1, sample/G2} area  
Performing get_lib_attribute on library object 'sample/G1'.  
Performing get_lib_attribute on library object 'sample/G2'.  
{1.0, 2.0}
```

SEE ALSO

`set_lib_attribute(2)`

get_lib_cells

Creates a collection of library cells from the libraries loaded into memory.

SYNTAX

```
collection get_lib_cells
  [-quiet]
  [-regexp | -exact]
  [-nocase]
  [-filter expression]
  [-scenarios scenario_names]
  patterns | -of_objects objects
```

Data Types

<i>expression</i>	string
<i>scenario_names</i>	list
<i>patterns</i>	list
<i>objects</i>	list

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding "." to the beginning or end of the expressions, as needed.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each library cell in the collection, the expression is evaluated based on the library cell's attributes. If the expression evaluates to true, the library cell is included in the result.

To see the list of library cell attributes that you can use in the expression, use the **list_attributes -application -class lib_cell** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-scenarios *scenario_names*

Searches the libraries that are used in the specified scenarios.

If you do not specify this option, all libraries that are loaded into memory are searched.

patterns

Creates a collection of library cells whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

The *patterns* and **-of_objects** arguments are mutually exclusive; you must specify one.

-of_objects *objects*

Creates a collection of library cells that are referenced by the specified cells or own the specified library pins. Each object is either a named library pin, a netlist cell, a library pin collection, or a netlist cell collection.

The *patterns* and **-of_objects** arguments are mutually exclusive; you must specify one.

DESCRIPTION

This command creates a collection of library cells from the libraries currently loaded into memory that match the specified criteria.

If no libraries have been loaded into memory, the tool loads the libraries specified in the **link_library** variable into memory the first time you run the **get_libs**, **get_lib_cells**, or **get_lib_pins** command.

If you use the **-scenarios** option, only libraries associated with the specified scenarios are included in the search.

The command returns a collection if any library cells match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

By default, this command uses information from all active scenarios. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following example queries all library cells that are in the misc_cmos library whose names begin with AN2. Although the output looks like a list, it is just a display.

```
prompt> get_lib_cells misc_cmos/AN2*  
{misc_cmos/AN2 misc_cmos/AN2P}
```

The following example shows one way to determine the library cell used by a particular cell instance:

```
prompt> get_lib_cells -of_objects [get_cells o_reg1]  
{misc_cmos/FD2}
```

SEE ALSO

- collections(2)
- filter_collection(2)
- get_cells(2)
- list_attributes(2)
- query_objects(2)
- collection_result_display_limit(3)
- wildcards(3)

get_lib_pins

Creates a collection of library cell pins from libraries loaded into memory.

SYNTAX

```
collection get_lib_pins
  [-quiet]
  [-regex | -exact]
  [-nocase]
  [-filter expression]
  patterns | -of_objects objects
```

Data Types

```
expression  string
patterns    list
objects     list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regex

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regex** command. When using the **-regex** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed.

The **-regex** and **-exact** options are mutually exclusive; you can use only one.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

The **-regex** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each library cell pin in the collection, the expression is evaluated based on the library cell pin's attributes. If the expression evaluates to true, the library cell pin is included in the result.

To see the list of library cell pin attributes that you can use in the expression, use the **list_attributes -application -class lib_pin** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

patterns

Creates a collection of library cell pins whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

The *patterns* and **-of_objects** arguments are mutually exclusive; you must specify one.

-of_objects *objects*

Creates a collection of library cell pins referenced by the specified netlist pins or owned by the specified library cells. Each object is either a named library cell, netlist pin, library cell collection, or a netlist pin collection.

The *patterns* and **-of_objects** arguments are mutually exclusive; you must specify one.

DESCRIPTION

This command creates a of library cell pins from the libraries currently loaded into memory that match the specified criteria.

If no libraries have been loaded into memory, the tool loads the libraries specified in the **link_library** variable into memory the first time you run the **get_libs**, **get_lib_cells**, or **get_lib_pins** command.

The command returns a collection if any library cell pins match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following example queries all pins of the AN2 library cell in the misc_cmos library. Although the output looks like a list, it is just a

display.

```
prompt> get_lib_pins misc_cmos/AN2/*  
{misc_cmos/AN2/A misc_cmos/AN2/B misc_cmos/AN2/Z}
```

The following example shows one way to find out how the library pin is used by a particular pin in the netlist:

```
prompt> get_lib_pins -of_objects o_reg1/Q  
{misc_cmos/FD2/Q}
```

SEE ALSO

- collections(2)
- filter_collection(2)
- get_libs(2)
- get_lib_cells(2)
- list_attributes(2)
- query_objects(2)
- collection_result_display_limit(3)
- wildcards(3)

get_libs

Creates a collection of libraries loaded into memory.

SYNTAX

```
collection get_libs
  [-quiet]
  [-regex | -exact]
  [-nocase]
  [-filter expression]
  [-scenarios scenario_names]
  [patterns | -of_objects objects]
```

Data Types

<i>expression</i>	string
<i>scenario_names</i>	list
<i>patterns</i>	list
<i>objects</i>	list

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regex

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regex** command. When using the **-regex** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding "." to the beginning or end of the expressions, as needed.

The **-regex** and **-exact** options are mutually exclusive; you can use only one.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

The **-regex** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each library in the collection, the expression is evaluated based on the library's attributes. If the expression evaluates to true, the library is included in the result.

To see the list of library attributes that you can use in the expression, use the **list_attributes -application -class library** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-scenarios *scenario_names*

Searches the libraries that are used in the specified scenarios.

If you do not specify this option, all libraries that are loaded into memory are searched.

patterns

Creates a collection of libraries whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses the * (asterisk) as the default pattern.

-of_objects *objects*

Creates a collection of libraries that contain the specified objects. Each object is either a named library cell or a library cell collection.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses the * (asterisk) as the default pattern.

DESCRIPTION

This command creates a collection of libraries from the libraries currently loaded into memory that match the specified criteria.

If no libraries have been loaded into memory, the tool loads the libraries specified in the **link_library** variable into memory the first time you run the **get_libs**, **get_lib_cells**, or **get_lib_pins** command.

If you use the **-scenarios** option, only libraries associated with the specified scenarios are included in the search.

The command returns a collection if any libraries match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

By default, this command uses information from all active scenarios. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following example queries all loaded libraries. Use the **list_libs** command to get a complete listing of the libraries.

```
prompt> get_libs  
{misc_cmos misc_cmos_io}
```

SEE ALSO

- [collections\(2\)](#)
- [filter_collection\(2\)](#)
- [list_attributes\(2\)](#)
- [list_libs\(2\)](#)
- [query_objects\(2\)](#)
- [regexp\(2\)](#)
- [collection_result_display_limit\(3\)](#)
- [wildcards\(3\)](#)

get_license

Obtains a license for a feature.

SYNTAX

```
status get_license
      feature_list
      [-quantity num_licenses]
```

Data Types

```
feature_list  list
num_licenses integer
```

ARGUMENTS

feature_list

Specifies a list of features to be obtained. If you specify more than one feature, the list must be enclosed in braces ({}).

By looking at your key file, you can determine all of the features licensed at your site.

-quantity *num_licenses*

Specifies the total number of licenses to be checked out for each feature after the command has completed. If some licenses have already been checked out, the command acquires only the additional licenses needed to bring the total to the specified quantity. If this option is not specified, only one license is checked out for each feature.

DESCRIPTION

This command obtains a license for the specified features. These features are checked out by the current user until the **remove_license** command is used or until the program exits.

If multiple licenses are required for a multicore run, you can use the **-quantity** option to specify the total number of licenses needed (which might be different than the incremental number of licenses that the command checks out). You can use this option to reserve the required number of licenses early in the session, rather than risking a failure later in the session if the licenses are not available.

The **list_licenses** command provides a list of the features that you are currently using.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example obtains the multivoltage feature:

```
prompt> get_license Galaxy-MV
```

The following example obtains two copies of the licenses required to run a multicore **compile_ultra** command in Design Compiler:

```
prompt> get_license -quantity 2 {DC-Expert DC-Ultra-Opt DC-Ultra-Features}
```

SEE ALSO

- check_license(2)
- license_users(2)
- list_licenses(2)
- remove_license(2)

get_magnet_cells

Returns a collection of magnet cells that can be pulled closer to the specified magnet objects.

SYNTAX

```
collection get_magnet_cells  
  magnet_objects  
  [-stop_by_sequential_cells]  
  [-exclude_buffers]  
  [-exclude_cells object_list]  
  [-logical_level level  
  | -stop_points object_list]
```

Data Types

```
magnet_objects  collection  
object_list     collection  
level           integer
```

ARGUMENTS

magnet_objects

Specifies the magnet objects, which can be fixed macro cells, a pin of a fixed macro cell, or I/O pins. Vias, blockages, and I/O pads are not allowed as magnets.

-stop_by_sequential_cells

Specifies to stop cell magnetization when the tool traverses the logical levels and encounters a sequential cell. By default, all cells in the logical levels of the specified magnet objects are magnetized by this command.

-exclude_buffers

Prevents counting buffers and inverters when calculating the level. Buffers and inverters are pulled like normal cells but are not used for level determination. By default, buffers and inverters are considered as one logic level.

-exclude_cells *object_list*

Specifies to exclude the specified cells when the command reports the target cells.

-logical_level *level*

Specifies the number of logical levels to magnetize. By default, the *level* value is 1; that is, only the first level cells are magnetized.

The **-logical_level** and **-stop_points** options are mutually exclusive.

-stop_points *object_list*

Identifies the cells that lie on the timing path between the specified magnet objects and the objects specified by this option. You can specify any pins, ports, or cell objects for this option.

The **-logical_level** and **-stop_points** options are mutually exclusive.

DESCRIPTION

The **get_magnet_cells** command returns a collection of cells that can be pulled toward the specified magnet objects. You specify magnet objects by using **get_magnet_cells** *magnet_objects*.

You can use the returned collection to highlight the cells affected by magnet placement in the GUI.

If there is a large number of high fanout nets, magnet placement might pull a large number of cells, which can result in long runtime. To avoid this, adjust the value of the **magnet_placement_fanout_limit** variable. The default is 1000.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns a collection of cells that are affected by magnet placement on a cell named INST_1:

```
prompt> get_magnet_cells [get_cells "INST_1"]
```

The following example returns a collection of cells that get pulled during magnet placement excluding a cell named INST_2. Two levels of cells are pulled. If any buffers exist, they are skipped.

```
prompt> get_magnet_cells \  
-exclude_buffers [get_cells "INST_2"]
```

SEE ALSO

magnet_placement(2)
set_cell_location(2)
magnet_placement_fanout_limit(3)

get_matching_nets_for_pattern

Gets all matching nets for a defined search pattern. The time units for the **-setup_***, **-hold_***, and **-transition_*** options should always be the same as the time unit returned by the **report_units** command.

SYNTAX

```
collection get_matching_nets_for_pattern
  -pattern id
  [-setup_slack_lower_limit number]
  [-setup_slack_upper_limit number]
  [-hold_slack_lower_limit number]
  [-hold_slack_upper_limit number]
  [-transition_lower_limit number]
  [-transition_upper_limit number]
  [-optimizable]
```

Data Types

id integer
number float

ARGUMENTS

-pattern *id*

Specifies a net pattern ID number. Net pattern ID numbers are created by the **create_net_search_pattern** command, which utilizes combinations of physical attributes to derive collections of nets.

This is a required option.

-setup_slack_lower_limit *number*

Identifies nets in the specified pattern ID that have a setup slack that is greater than or equal to the specified value. A net's setup slack is the worst setup slack seen across all its sinks. To select all of the nets in pattern 1 that have a setup slack of more than 100, use **get_matching_nets_for_pattern -setup_slack_lower_limit 100 -pattern 1**.

-setup_slack_upper_limit *number*

Identifies nets in the specified pattern ID that have a setup slack that is less than the specified value. A net's setup slack is the worst setup slack seen across all its sinks. To select all of the nets in pattern 2 that have a setup slack of less than 0, use **get_matching_nets_for_pattern -setup_slack_upper_limit 0 -pattern 2**.

-hold_slack_lower_limit *number*

Identifies nets in the specified pattern ID that have a hold slack that is greater than or equal to the specified value. A net's hold slack is the worst hold slack seen across all its sinks. To select all of the nets in pattern 3 that have a hold slack of more than

1000, use **get_matching_nets_for_pattern -hold_slack_upper_limit 1000 -pattern 3**.

-hold_slack_upper_limit *number*

Identifies nets in the specified pattern ID that have a hold slack that is less than the specified value. A net's hold slack is the worst hold slack seen across all its sinks. To select all of the nets in pattern 4 that have a hold slack of less than 0, use **get_matching_nets_for_pattern -hold_slack_upper_limit 0 -pattern 4**.

-transition_lower_limit *number*

Identifies nets in the specified pattern ID that have a net transition time that is greater than or equal to the specified value. A net's transition time is the worst transition time seen across all its sinks. To select all of the nets in pattern 5 that have a transition time of more than 100, use **get_matching_nets_for_pattern -transition_lower_limit 100 -pattern 5**.

-transition_upper_limit *number*

Identifies nets in the specified pattern ID that have a transition time that is less than the specified value. A net's transition time is the worst transition time seen across all its sinks. To select all of the nets in pattern 6 that have a transition time of less than 90, use **get_matching_nets_for_pattern -transition_upper_limit 90 -pattern 6**.

-optimizable

Reduces a collection of pattern-matched nets to include only nets that can be optimized. When working with patterns, it is best to ignore clock nets, ideal nets, constant nets, nets with multiple drivers, tristate nets, dont_touch nets, DRC disabled nets, and nets with no driver. For example, if you want to create buffer trees from a subset of all nets that exist in pattern 7, you can use **create_buffer_tree -from [get_matching_nets_for_pattern -optimizable -pattern 7]**.

DESCRIPTION

The **get_matching_nets_for_pattern** command returns a collection of nets that match a predefined pattern. The collection of nets it returns can be redirected to pattern-independent commands, such as **change_selection**, **create_buffer_tree**, and **set_net_routing_layer_constraints**.

Multicorner-Multimode Support

This command uses information from all active scenarios. For setup slack, hold slack, and transition time, the worst value from all active scenarios is always used.

EXAMPLES

Here are some examples of how the **get_matching_nets_for_pattern** command can be used:

```
prompt> get_matching_nets_for_pattern -pattern 3
prompt> sizeof_collection [get_matching_nets_for_pattern -pattern 3]

prompt> set_net_routing_layer_constraints \
-min_layer_name M7 -max_layer_name M10 \
[get_matching_nets_for_pattern -optimizable -pattern 3]
```

SEE ALSO

create_net_search_pattern(2)
report_net_search_pattern(2)
remove_net_search_pattern(2)
set_net_search_pattern_delay_estimation_options(2)
report_net_search_pattern_delay_estimation_options(2)
set_net_search_pattern_priority(2)
report_net_search_pattern_priority(2)

get_message_ids

Get application message ids

SYNTAX

```
string get_message_ids
    [-type severity]
    [pattern]
```

Data Types

<i>severity</i>	string
<i>pattern</i>	string

ARGUMENTS

-type *severity*

Filter ids based on type (Values: Info, Warning, Error, Severe, Fatal)

pattern

Get IDs matching pattern (default: *)

DESCRIPTION

The **get_message_ids** command retrieves the error, warning and informational messages used by the application. The result of this command is a Tcl formatted list of all message ids. Information about the id can be queried with the **get_message_info** command.

EXAMPLES

The following code finds all error messages and makes the application stop script execution when one of these messages is encountered.

```
foreach id [get_message_ids -type Error] {
    set_message_info -stop_on -id $id
}
```

SEE ALSO

`print_message_info(2)`
`set_message_info(2)`
`suppress_message(2)`

get_message_info

Returns information about diagnostic messages.

SYNTAX

```
integer get_message_info  
[-error_count  
|-warning_count  
|-info_count  
|-limit l_id  
|-occurrences o_id  
|-suppressed s_id  
|-id i_id]
```

Data Types

```
l_id  string  
o_id  string  
s_id  string  
i_id  string
```

ARGUMENTS

-error_count

Returns the number of error messages issued so far.

-warning_count

Returns the number of warning messages issued so far.

-info_count

Returns the number of informational messages issued so far.

-limit *l_id*

Returns the current user-specified limit for a given message ID. The limit was set with the **set_message_info** command.

-occurrences *o_id*

Returns the number of occurrences of a given message ID.

-suppressed *s_id*

Returns the number of times a message was suppressed either using **suppress_message** or due to exceeding a user-specified

limit.

-id *i_id*

Returns information about the specified message. The information is returned as a Tcl list compatible with the array set command.

DESCRIPTION

The **get_message_info** command retrieves information about error, warning, and informational messages. For example, if the following message is generated, information about it is recorded:

Error: unknown command 'wrong_command' (CMD-005)

It is useful to be able to retrieve recorded information about generated diagnostic messages. For example, you can stop a script after a certain number of errors have occurred, or monitor the number of messages issued by a single command.

You can also find out how many times a specific message has occurred, or how many times it has been suppressed. Also, you can find out if a limit has been set for a particular message ID.

EXAMPLES

The following example uses the **get_message_info** command to count the number of errors that occurred during execution of a specific command, and to return from the procedure if the error count exceeds a given amount:

```
prompt> proc \
do_command {limit} {
    set current_errors [get_message_info -error_count]
    command
    set new_errors [get_message_info -error_count]
    if {[expr $new_errors - $current_errors] > $limit} {
        return -code error "Too many errors"
    }
    ...
}
```

The following example uses the **get_message_info** command to retrieve information on the CMD-014 message:

```
prompt> get_message_info -id CMD-014
id CMD-014 severity Error limit 0 occurrences 0 suppressed 0 message
{Invalid %s value '%s' in list.}
```

SEE ALSO

print_message_info(2)
set_message_info(2)
suppress_message(2)
get_message_ids(2)

get_multibits

Creates a collection of one or more multibits loaded into `dc_shell`. You can assign these multibits to a variable or pass them into another command.

SYNTAX

```
string get_multibits  
  [-quiet]  
  [-regex]  
  [-nocase]  
  [-filter expression]  
  [patterns]
```

Data Types

```
expression  string  
patterns    list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regex

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-filter *expression*

Filters the collection with *expression*. For any multibits that match *patterns*, the expression is evaluated based on the multibit's attributes. If the expression evaluates to true, the multibit is included in the result. This option works the same as the **filter** command in `dc_shell`.

patterns

Matches multibit names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards, see the **wildcards** man page.

DESCRIPTION

The **get_multibits** command creates a collection of multibits from those currently loaded into dc_shell that match certain criteria. The command returns a collection handle (identifier) if any multibits match the *patterns* and pass the filter (if specified). If no objects matched your criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding "." to the beginning or end of the expressions as needed.

You can use the **get_multibits** command at the command prompt, or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the **get_multibits** result to a variable.

When issued from the command prompt, **get_multibits** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum using the **collection_result_display_limit** variable.

The implicit query property of **get_multibits** provides a fast, simple way to display multibits in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_multibits** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the multibits that begin with *out*:

```
prompt> get_multibits out*
{out_reg}
```

The following example queries the multibits that begin with *test* in the design instance *mid1/bot1*:

```
prompt> get_multibits mid1/bot1/test*
{mid1/bot1/test_reg}
```

SEE ALSO

collections(2)
filter(2)
filter_collection(2)
query_objects(2)
collection_result_display_limit(3)
wildcards(3)

get_nets

Creates a collection of nets that match the specified criteria.

SYNTAX

```
collection get_nets
  [-quiet]
  [-regex | -exact]
  [-nocase]
  [-filter expression]
  [-hierarchical]
  [-top_net_of_hierarchical_group]
  [-segments]
  [-all]
  [-boundary_type upper | lower | both]
  [-rtl]
  [patterns | -of_objects objects]
```

Data Types

```
expression  string
patterns    list
objects     collection
```

ARGUMENTS

-quiet

Suppresses messages if no objects match. Syntax error messages are not suppressed.

-regex

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regex** command. When using the **-regex** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding "." to the beginning or end of the expressions, as needed.

The **-regex** and **-exact** options are mutually exclusive; you can use only one.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the value of the *expression* argument. For any nets that match the specified criteria, the expression is evaluated based on the net's attributes. If the expression evaluates to true, the net is included in the result.

-hierarchical

Searches for nets level-by-level relative to the current instance. The name of the object at a particular level must match the patterns. For example, if there is a net named block1/muxsel, a hierarchical search finds it using muxsel.

You cannot use the **-hierarchical** option with the **-of_objects** option.

-top_net_of_hierarchical_group

Keeps only the top net of a hierarchical group. When more than one hierarchical net of the same group is specified (local nets at various hierarchical levels of the same physical net), only the net closest to the top of the hierarchy is saved in the collection. In the case of multiple nets at the same level, the first net specified is kept. To get the top hierarchical net connected to a single net, need to use this option in combination with the **-segments** option.

-segments

Modifies the initial search that matches the *patterns* or **-of_object** argument to include all global segments for the matching nets. Global net segments are all the net segments that are physically connected across all hierarchical boundaries.

This option is best used with a single net.

When you use the **-segments** option with the **-top_net_of_hierarchical_group** option, you can isolate the highest net segment of a physical net.

-all

Includes power, ground, and tie nets.

-boundary_type upper | lower | both

Specifies what to do when getting nets of boundary pins.

You must specify one of the following values:

- **upper** (default)
Get the net outside of the hierarchical block.
- **lower**
Get the net inside of the hierarchical block.
- **both**
Get the net both inside and outside of the hierarchical block.

This option can be used only with the **-of_objects** option. In addition, the specified object must be a hierarchical pin or a pin on a block abstraction. This option has no meaning for pins that are not hierarchical and are not on block abstractions.

-rtl

Creates a collection of nets with the **object_rtl_name** attribute.

patterns

Creates a collection of nets whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

The *patterns* and **-of_objects** options arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

-of_objects *objects*

Creates a collection of nets connected to the specified objects. Each object can be a pin, port, or cell.

The *patterns* and **-of_objects** options arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

You cannot use the **-hierarchical** option with the **-of_objects** option.

DESCRIPTION

This command creates a collection of nets in the current design relative to the current instance that match the specified criteria.

If the *patterns* and **-of_objects** arguments fail to match any objects and the current design is not linked, the design automatically links.

The command returns a collection if any nets match the specified criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the nets that begin with NET in a block named block1. Although the output looks like a list, it is a display.

```
prompt> get_nets block1/NET*
{block1/NET1QNX block1/NET2QNX}
```

The following example queries the nets connected to a collection of pins.

```

prompt> current_instance block1
block1
prompt> set pinsel [get_pins {o_reg1/QN o_reg2/QN}]
{o_reg1/QN o_reg2/QN}
prompt> get_nets -of_objects $pinsel
{NET1QNX NET2QNX}

```

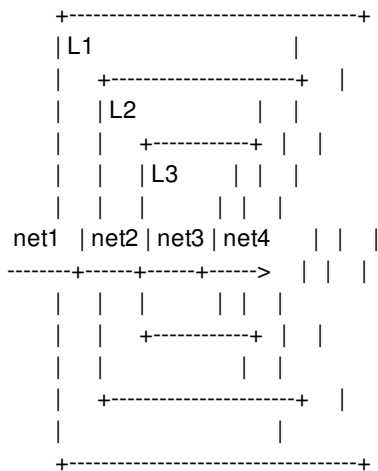
The following example queries the nets connected to a collection of cells.

```

prompt> current_instance block1
block1
prompt> set cellsel [get_cells {o_reg1 o_reg2}]
{o_reg1 o_reg2}
prompt> get_nets -of_objects $cellsel
{NET1QX NET1QNX NET1DX NET2QX NET2QNX NET2DX}

```

The following examples use the design shown below to show the behavior of the **get_nets** command with various options. There is a buffer instance named **buffer** in the **L3** instance.



When you use the **-of_objects** option by itself, the command gets only the net that connects to a pin directly.

```

prompt> get_nets -of_objects L1/L2/L3/buffer/A
{L1/L2/L3/net_4}

```

When you also use the **-segments** option, the command gets all the hierarchical nets that connect together through the hierarchical pins.

```

prompt> get_nets -of_objects L1/L2/L3/buffer/A -segments
{L1/L2/L3/net_4 L1/L2/net_3 L1/net_2 net_1}

```

When you use the **-segments** and **-top_net_of_hierarchical_group** option together, the command returns only the net in the topmost hierarchical net group.

```

prompt> get_nets -of_objects L1/L2/L3/buffer/A -segments \  

-top_net_of_hierarchical_group  

{net_1}

```

The following examples use the *patterns* argument to select the nets.

```

prompt> get_nets net*4 -hierarchical
{L1/L2/L3/net_4}

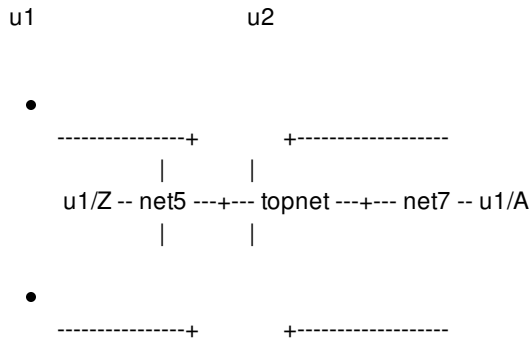
prompt> get_nets net*4 -hierarchical -segments

```

```
{L1/L2/net_3 net_1 L1/L2/L3/net_4 L1/net_2}
```

```
prompt> get_net net*4 -hierarchical -segments -filter name==net_2
{L1/net_2}
```

This example shows how to use the *-boundary_type* options. Given the following circuit:



there are hierarchical cells u1 and u2 at this level, connected by a net *topnet*. Within u1 is a pin u1/Z driving *net5*, and within u2 is a pin u1/A being driven by *net7*. These three nets are physically the same net. Assume these are the only nets in the design. Notice the difference in the results of the following two **get_nets** commands:

```
prompt> get_nets * -hierarchical
{topnet u1/net5 u2/net7}
prompt> get_nets * -hierarchical -top_net_of_hierarchical_group
{topnet}
```

Assume that at the top level, topnet is connected to pins u2/in and u1/out. Here are some examples of the *-boundary_type* option. Notice now in this case, the "upper" type does not add value.

```
prompt> get_nets -of_objects u2/in
{topnet}
prompt> get_nets -of_objects u2/in -boundary_type upper
{topnet}
prompt> get_nets -of_objects u2/in -boundary_type lower
{u2/net7}
prompt> get_nets -of_objects u2/in -boundary_type both
{u2/net7 topnet}
prompt> get_nets -boundary lower -of_objects \
    [get_pins -of_objects [get_nets topnet]]
{u1/net5 u2/net7}
```

SEE ALSO

```
collections(2)
filter_collection(2)
list_attributes(2)
query_objects(2)
collection_result_display_limit(3)
wildcards(3)
```

get_object_name

Returns a list of names of the objects in a collection.

SYNTAX

```
list get_object_name  
    collection
```

Data Types

collection string

ARGUMENTS

collection

Specifies the name of the collection that contains objects whose names are requested.

DESCRIPTION

This command returns a list of names of the objects in a collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns the name top as the object contained in the collection returned by the **current_design** command.

```
prompt> get_object_name [current_design]  
Current design is 'top'.  
top
```

SEE ALSO

`collections(2)`

get_path_groups

Creates a collection of path groups that match the specified criteria.

SYNTAX

```
collection get_path_groups
  [-quiet]
  [-regexp]
  [-nocase]
  [-filter expression]
  patterns
```

Data Types

```
expression  string
patterns    list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the `==` and `!~` filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding `".*"` to the beginning or end of the expressions, as needed.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the `==`, `==~`, and `!~` filter operators.

-filter *expression*

Filters the collection with the specified expression. For each path group in the collection, the expression is evaluated based on the path group's attributes. If the expression evaluates to true, the path group is included in the result.

For information about the plan group attributes, see SolvNet article 008622.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

patterns

Creates a collection of path groups whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

DESCRIPTION

This command creates a collection of path groups from the current design that match certain criteria. Path groups control the optimization cost function and also affect timing reports.

The command returns a collection if any path groups match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command uses information from the current scenario only.

SEE ALSO

collections(2)
filter_collection(2)
foreach_in_collection(2)
group_path(2)
query_objects(2)
report_path_group(2)
collection_result_display_limit(3)
wildcards(3)

get_physical_hierarchy

Returns a list of hierarchical cells that are physical hierarchies in the current design.

SYNTAX

integer **get_physical_hierarchy**

ARGUMENTS

The `\bget_physical_hierarchy` command has no arguments.

DESCRIPTION

The **get_physical_hierarchy** command returns a list of names of hierarchical cells in the current design that are physical hierarchies. These hierarchical cells are set as physical hierarchies through the **set_physical_hierarchy** command.

EXAMPLES

The following example first shows how to set the *U1* and *U2* hierarchical cells as physical hierarchies and then shows how they are returned by the **get_physical_hierarchy** command:

```
prompt> set_physical_hierarchy {U1 U2}
```

```
prompt> get_physical_hierarchy  
{ U1, U2 }
```

SEE ALSO

`set_physical_hierarchy(2)`

get_pins

Creates a collection of pins that match specified criteria.

SYNTAX

```
collection get_pins
  [-quiet]
  [-regexp | -exact]
  [-nocase]
  [-filter expression]
  [-hierarchical]
  [patterns | -of_objects objects [-leaf]]
```

Data Types

```
expression  string
patterns    list
objects     collection
```

ARGUMENTS

-quiet

Suppresses warning and error messages when no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

The **-regexp** and **-exact** options are mutually exclusive; you cannot use both.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each pin in the collection, the expression is evaluated based on the pin's attributes. If the expression evaluates to true, the pin is included in the result.

To see the list of pin attributes that you can use in the expression, use the **list_attributes -application -class pin** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-hierarchical

Searches for pins level-by-level, relative to the current instance. The name of the object at a particular level must match the patterns. The search is similar to that of the UNIX **find** command. For example, if there is a pin named block1/adder/D[0], a hierarchical search finds it by using adder/D[0].

The **-hierarchical** option cannot be used with the **-of_objects** option.

patterns

Creates a collection of pins whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case-sensitive unless you use the **-nocase** option.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify no more than one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

-of_objects *objects*

Creates a collection of pins connected to the specified objects. Each object can be a cell or net.

By default, the command considers only pins connected to the specified nets at the same hierarchical level. To consider only pins connected to leaf cells on the specified nets, use the **-leaf** option.

You cannot use the **-hierarchical** option with the **-of_objects** option.

-leaf

Includes only those pins that are on leaf cells connected to the nets specified in the **-of_objects** option. The command crosses hierarchical boundaries to find pins on leaf cells.

You can use this option only if you also use the **-of_objects** option.

DESCRIPTION

This command creates a collection of pins in the current design relative to the current instance that match specified criteria.

If the *patterns* and *objects* arguments do not match any objects and the current design is not linked, the command automatically links the design.

The command returns a collection if any pins match the specified criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**, or assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report

the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by setting the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the CP pins of cells that begin with the letter "o". Although the output looks like a list, it is only a text report.

```
prompt> get_pins o*/CP
{o_reg1/CP o_reg2/CP o_reg3/CP o_reg4/CP}
```

The following example queries the pins connected to a collection of cells:

```
prompt> set csel [get_cells o_reg1]
{o_reg1}

prompt> get_pins -of_objects $csel
{o_reg1/D o_reg1/CP o_reg1/CD o_reg1/Q o_reg1/QN}
```

The following example shows the difference between getting the local pins of a net and the leaf pins of net. In this example, NET1 is connected to the i2/aP and reg1/QN. Cell i2 is hierarchical. Within cell i2, port a is connected to U1/A and U2/A.

```
prompt> get_pins -of_objects [get_nets NET1]
{i2/a reg1/QN}

prompt> get_pins -leaf -of_objects [get_nets NET1]
{i2/U1/A i2/U2/A reg1/QN}
```

The following example shows how to create a clock using a collection of pins:

```
prompt> create_clock -period 8 -name CLK [get_pins o_reg*/CP]
1
```

SEE ALSO

collections(2)
 filter_collection(2)
 list_attributes(2)
 query_objects(2)
 collection_result_display_limit(3)
 wildcards(3)

get_placement_area

Returns a list of coordinates for the current core placement area.

SYNTAX

string **get_placement_area**

ARGUMENTS

None.

DESCRIPTION

Returns a list of coordinates for the current core placement area. Returns a list of float numbers in the order of lower-left corner coordinates and upper-right corner coordinates. If the core area is not defined in the current design, the command returns an error message.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to get the core placement area coordinates in the current design.

```
prompt> get_placement_area
```

SEE ALSO

set_placement_area(2)

get_placement_blockages

Creates a collection of placement blockages that match the specified criteria.

SYNTAX

```
collection get_placement_blockages
  [-quiet]
  [-within region
  | -touching region
  | -intersect region]
  [-filter expression]
  [-type hard | soft | pin | hard_macro | partial]
  [patterns]
```

Data Types

region list of points
expression string
patterns list

ARGUMENTS

-quiet

Suppresses warning and error messages.

-within *region*

Creates a collection that contains all placement blockages that are completely inside the specified region and do not overlap the boundary. The region boundary can be a rectangle or a polygon.

The format for specifying a rectangle is $\{\{lx\ ly\} \{urx\ ury\}\}$ or $\{lx\ ly\ urx\ ury\}$, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is $\{\{x1\ y1\} \{x2\ y2\} \dots \{xN\ yN\} \{x1\ y1\}\}$, where each $\{x\ y\}$ pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, and **-intersect** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the command searches the entire design.

-touching *region*

Creates a collection that contains all placement blockages that are inside the specified region, including those that overlap the boundary. The region boundary can be a rectangle or a polygon.

The format for specifying a rectangle is $\{\{lx\ ly\} \{urx\ ury\}\}$ or $\{lx\ ly\ urx\ ury\}$, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is $\{\{x1\ y1\} \{x2\ y2\} \dots \{xN\ yN\} \{x1\ y1\}\}$, where each $\{x\ y\}$ pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, and **-intersect** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the command searches the entire design.

-intersect region

Creates a collection that contains all placement blockages that intersect the boundary of the specified region and at least part of the cell is outside of the specified region. The region boundary can be a rectangle or a polygon.

The format for specifying a rectangle is $\{\{lx\ ly\} \{urx\ ury\}\}$ or $\{lx\ ly\ urx\ ury\}$, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is $\{\{x1\ y1\} \{x2\ y2\} \dots \{xN\ yN\} \{x1\ y1\}\}$, where each $\{x\ y\}$ pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, and **-intersect** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the command searches the entire design.

-filter expression

Filters the collection with the specified expression. For each placement blockage in the collection, the expression is evaluated based on the placement blockage's attributes. If the expression evaluates to true, the placement blockage is included in the result.

To see the list of placement blockage attributes that you can use in the expression, use the **list_attributes -application -class placement_blockage** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-type hard | soft | pin | hard_macro | partial

Indicates the type of placement blockage. By default, this command matches all kinds of placement blockages.

patterns

Creates a collection of placement blockages whose names match the specified patterns. Placement blockages that are named by the tool have the naming convention PB#*n*, where *n* is an integer. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive.

If you do not specify this argument, the command uses * (asterisk) as the default pattern.

DESCRIPTION

This command creates a collection of placement blockages by selecting placement blockages from the current design that match the specified criteria.

The command returns a collection if any ports match the criteria. If no objects match the criteria, the command returns an empty

string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples show some uses of this command to create collections of placement blockages:

```
prompt> get_placement_blockages -within {{2 2} {25 25}}
{PB#5389}
```

```
prompt> get_placement_blockages -filter "area > 1000"
{PB#4683}
```

SEE ALSO

collections(2)
create_placement_blockage(2)
filter_collection(2)
get_attribute(2)
list_attributes(2)
query_objects(2)
remove_placement_blockage(2)
remove_route_guide(2)
collection_result_display_limit(3)
wildcards(3)

get_polygon_area

Calculate the area of the input polygon.

SYNTAX

```
double get_polygon_area  
    polygon
```

Data Types

polygon list

ARGUMENTS

polygon

A list of points that represents a polygon and the format is like this: {{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}. The coordinate unit is specified in technology file (usually it is micron).

DESCRIPTION

This command returns the area covered by the input polygon. The input for this command is a list of points which represents a polygon; and the returned value represents the area of the input polygons. Before this command is used, the library should be opened.

EXAMPLES

The following command returns a value represents the area of the input polygons.

```
prompt> get_polygon_area {{5 5} {20 5} {20 20} {15 20} {15 10} \  
    {10 10} {10 15} {5 15} {5 5}}  
150.0
```

SEE ALSO

convert_to_polygon(2)
resize_polygon(2)

get_ports

Creates a collection of ports from the current design that match the specified criteria.

SYNTAX

```
collection get_ports
  [-quiet]
  [-regexp | -exact]
  [-nocase]
  [-filter expression]
  [-hierarchical]
  [patterns | -of_objects objects]
```

Data Types

```
expression  string
patterns    list
objects     collection
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding "." to the beginning or end of the expressions, as needed.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each port in the collection, the expression is evaluated based on the port's attributes. If the expression evaluates to true, the port is included in the result.

To see the list of port attributes that you can use in the expression, use the **list_attributes -application -class port** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-hierarchical

Searches for ports level-by-level, relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to that of the UNIX **find** command. For example, if there is a port named *block1/adder/D[0]*, a hierarchical search finds it by using *adder/D[0]*.

The **-hierarchical** option is mutually exclusive with use the **-of_objects** option; you can use only one.

patterns

Creates a collection of ports whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

-of_objects *objects*

Creates a collection of ports connected to the specified objects. Each object can be a net, terminal, bound, or via region.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

DESCRIPTION

This command creates a collection of ports by selecting ports from the current design that match the specified criteria.

The command returns a collection if any ports match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

In addition, see the man pages for the **all_inputs** and **all_outputs** commands, which also create collections of ports.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries all input ports beginning with "mode". Although the output looks like a list, it is only a display.

```
prompt> get_ports mode* -filter {@port_direction == in}  
{mode[0] mode[1] mode[2]}
```

The following example sets the driving cell for ports beginning with "in" to an FD2 library cell.

```
prompt> set_driving_cell -lib_cell FD2 -library my_lib \  
[get_ports in*]
```

The following example reports ports connected to nets that match the pattern "bidir*".

```
prompt> report_port [get_ports -of_objects [get_nets bidir*]]
```

The following example get the ports connected to terminals that match the pattern "CC*".

```
prompt> get_ports -of_objects [get_terminals CC*]  
{CC CCEN}
```

The following example shows you can get bus ports by their base name. A[0], A[1], and A[2] are bus ports with the base name A; A[3] is not a bus port.

```
prompt> get_ports A  
{A[0] A[1] A[2]}  
  
prompt> get_ports A[3]  
{A[3]}
```

SEE ALSO

- all_inputs(2)
- all_outputs(2)
- collections(2)
- filter_collection(2)
- list_attributes(2)
- query_objects(2)
- collection_result_display_limit(3)
- find_allow_only_non_hier_ports(3)
- wildcards(3)

get_power_domains

Creates a collection of power domains that match the specified criteria.

SYNTAX

```
collection get_power_domains
  [-quiet]
  [-regex]
  [-nocase]
  [-filter expression]
  [-hierarchical]
  [patterns | -of_objects objects]
```

Data Types

```
expression  string
patterns    list
objects     collection
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regex

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern. This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns. The regular expression matching is similar to the Tcl **regex** command. When using the **-regex** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each power domain in the collection, the expression is evaluated based on the power domain's attributes. If the expression evaluates to true, the power domain is included in the result.

To see the list of power domain attributes that you can use in the expression, use the **list_attributes -application -class power_domain** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-hierarchical

Searches for power domains in the current scope and all of its descendant scopes.

If you specify this option, you can specify only simple names in the *patterns* argument; the name cannot contain hierarchy delimiter characters.

If you do not specify this option, the tool searches for power domains only in the current scope and you can use hierarchy delimiter characters in the *patterns* argument to specify hierarchical power domain names relative to the current scope.

You cannot use this option with the **-of_objects** option.

patterns

Creates a collection of power domains whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify either of these arguments, the command uses * (asterisk) as the default pattern.

-of_objects *objects*

Creates a collection of power domains connected to the specified objects. The objects can be cells, supply nets, supply ports, or power switches.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify either of these arguments, the command uses * (asterisk) as the default pattern.

DESCRIPTION

This command creates a collection of power domains by selecting power domains from the current design that match the specified criteria.

If no power domains match the criteria and the design is not linked, the tool automatically links the design.

The command returns a collection if any power domains match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets all power domains in the current scope:

```
prompt> get_power_domains -hierarchical  
{PD1 SUB_INST/PD2}
```

SEE ALSO

- `collections(2)`
- `filter_collection(2)`
- `list_attributes(2)`
- `query_objects(2)`
- `create_power_domain(2)`
- `remove_power_domain(2)`
- `report_power_domain(2)`
- `collection_result_display_limit(3)`
- `wildcards(3)`

get_power_switches

Creates a collection of power switches that match the specified criteria.

SYNTAX

```
collection get_power_switches
  [-quiet]
  [-regex]
  [-nocase]
  [-filter expression]
  [-hierarchical]
  [patterns]
```

Data Types

```
expression  string
patterns    list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regex

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the `==` and `!~` filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regex** command. When using the **-regex** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding `.*` to the beginning or end of the expressions, as needed.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the `==`, `==~`, and `!~` filter operators.

-filter *expression*

Filters the collection with the specified expression. For each power switch in the collection, the expression is evaluated based on the power switch's attributes. If the expression evaluates to true, the power switch is included in the result.

To see the list of power switch attributes that you can use in the expression, use the **list_attributes -application -class power_switch** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-hierarchical

Searches for power switches in the current scope and all of its descendant scopes.

If you specify this option, you can specify only simple names in the *patterns* argument; the name cannot contain hierarchy delimiter characters.

If you do not specify this option, the tool searches for power switches only in the current scope and you can use hierarchy delimiter characters in the *patterns* argument to specify hierarchical power switches relative to the current scope.

patterns

Creates a collection of power switches whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

If you do not specify this argument, the command uses * (asterisk) as the default pattern.

DESCRIPTION

This command creates a collection of power switches by selecting power switches from the current design that match the specified criteria.

If no power switches match the criteria and the design is not linked, the tool automatically links the design.

The command returns a collection if any power switches match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets all power switches under the current scope:

```
prompt> get_power_switches -hierarchical  
{SW1 SUB_INST/SW2}
```

SEE ALSO

`create_power_switch(2)`
`report_power_switch(2)`
`collections(2)`
`filter_collection(2)`
`list_attributes(2)`
`query_objects(2)`
`collection_result_display_limit(3)`
`wildcards(3)`

get_references

Creates a collection of one or more references loaded into memory.

SYNTAX

```
collection get_references
  [-hierarchical]
  [-quiet]
  [-regex]
  [-nocase]
  [-exact]
  [-filter expression]
  patterns
```

Data Types

```
expression  string
patterns    list
```

ARGUMENTS

-hierarchical

Searches for hierarchical cells whose references match the pattern. The full name of the object at a particular level must match the patterns. The use of this option does not force an autolink.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regex

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * (asterisk) and ? (question mark) wildcard characters.

-filter *expression*

Filters the collection with *expression*. For any references that match *patterns*, the expression is evaluated based on the attributes

of the reference. If the expression evaluates to true, the design is included in the result. This option works the same as the **filter** command in **dc_shell**.

patterns

Matches reference names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards, see the **wildcards** man page.

DESCRIPTION

The **get_references** command creates a collection of cells whose references match certain criteria. If no patterns match your criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can expand the search by adding `".*"` to the beginning or end of the expressions as needed.

You can use the **get_references** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_references** result to a variable.

When issued from the command prompt, **get_references** behaves as though **query_objects** has been executed to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum using the **collection_result_display_limit** variable.

The implicit query property of **get_references** provides a fast, simple way to display references in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_references** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the references that begin with *mpu*. Although the output looks like a list, it is just a display.

```
prompt> get_references mpu*
{mpu_0_0 mpu_0_1 mpu_1_0 mpu_1_1}
```

SEE ALSO

collections(2)
filter_collection(2)
query_objects(2)
collection_result_display_limit(3)

wildcards(3)

get_related_supply_net

Creates a collection of related supply nets of pins.

SYNTAX

```
collection get_related_supply_net  
  [pins]  
  [-ground]
```

Data Types

pins collection

ARGUMENTS

pins

Specifies the collection of pins to be queried.

-ground

Specifies that the ground supply net should be returned.

DESCRIPTION

The **get_related_supply_net** command creates a collection of supply nets of the specified collection of pins. Power supply net is returned by default. If the **-ground** option is specified, ground supply net is returned.

You can use the **get_related_supply_net** command either at the command prompt, or nest it as an argument to another command, such as the **report_supply_net** command. In addition, you can assign the result of the **get_related_supply_nets** command to a variable.

When you run the **get_related_supply_net** command at the prompt, it behaves as though the **query_objects** command has been called to display the objects in the collection. By default, a maximum of 100 objects are displayed. You can change this maximum limit by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the man page of the **collections** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets the supply nets of all pins of a level-shifter cell.

```
prompt> get_related_supply_net [get_pin Multiplier/mult_on_UPF_LS/*]  
{VDDL VDDL}
```

SEE ALSO

set_related_supply_net(2)
connect_supply_net(2)
report_supply_net(2)

get_route_zrt_common_options

Gets the value of the specified common route option.

SYNTAX

```
status get_route_zrt_common_options
      -name name_of_option
```

Data Types

name_of_option string

ARGUMENTS

-name *name_of_option*

Specifies the option whose value is returned. This is a required option.

DESCRIPTION

This command gets the value of the specified common route option. If you want to get the values of all the common route options, use the **report_route_zrt_common_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command returns the Zroute verbose level.

```
prompt> get_route_zrt_common_options -name verbose_level
```

SEE ALSO

```
set_route_zrt_common_options(2)  
report_route_zrt_common_options(2)
```

get_route_zrt_global_options

Gets the value of the specified global route option.

SYNTAX

status **get_route_zrt_global_options**
-name *name_of_option*

Data Types

name_of_option string

ARGUMENTS

-name *name_of_option*

Specifies the option whose value is returned. This is a required option.

DESCRIPTION

This command gets the value of the specified global route option. If you want to get the values of all the global route options, use the **report_route_zrt_global_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command tells whether crosstalk-driven global routing is enabled.

```
prompt> get_route_zrt_global_options -name crosstalk_driven
```

SEE ALSO

```
set_route_zrt_global_options(2)  
report_route_zrt_global_options(2)
```

get_rp_groups

Creates a collection of relative placement groups that match the specified criteria.

SYNTAX

```
collection get_rp_groups
  [patterns | -of_objects objects]
  [-quiet]
  [-nocase]
  [-exact]
  [-regex]
  [-ignored]
  [-top]
```

Data Types

```
patterns    list
objects     collection
```

ARGUMENTS

patterns

Creates a collection of relative placement groups whose names match the specified patterns. The format of each pattern string can be either *design_name::rp_group_name* or *rp_group_name*.

Both *design_name* and *rp_group_name* can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify either of these arguments, the command uses * (asterisk) as the default pattern.

-of_objects *objects*

Creates a collection of relative placement groups that contain the specified objects. Each object can be a cell, an instantiated or included relative placement group or a relative placement group keepout.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify either of these arguments, the command uses * (asterisk) as the default pattern.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-nocase

Makes matches case-insensitive.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding `".*"` to the beginning or end of the expressions, as needed.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-ignored

Includes only ignored relative placement groups.

-top

Includes only top-level relative placement groups.

DESCRIPTION

This command creates a collection of relative placement groups by selecting relative placement groups from the current design that match the specified criteria. This command is supported only for designs that do not contain multiply-instantiated designs.

The command returns a collection if any relative placement groups match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Relative placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the relative placement groups starting with "g" in a design starting with "r". Although the output looks like a list, the output is only a display.

```
prompt> get_rp_groups r*::g*  
{ripple::grp_ripple}
```

The following example uses the **get_rp_groups** command to set the utilization attribute of the relative placement groups.

```
prompt> set_rp_group_options [get_rp_groups r*::g*] \  
  -utilization 0.95  
{ripple::grp_ripple}
```

The following example queries the relative placement groups that include the U17 cell or the example3::block2 relative placement group.

```
prompt> get_rp_groups -of_objects {U17 example3::block2}  
{example3::gp_2_in_example3 example3::top_group}
```

SEE ALSO

- add_to_rp_group(2)
- all_rp_groups(2)
- create_rp_group(2)
- remove_from_rp_group(2)
- set_rp_group_options(2)
- write_rp_groups(2)
- collections(2)
- list_attributes(2)
- query_objects(2)
- collection_result_display_limit(3)
- wildcards(3)

get_scan_cell_names

Returns a list containing the names of scan cells for the specified option filters.

SYNTAX

```
list get_scan_cell_names
  [-view spec | existing_dft]
  [-chain chain_name_list | all]
  [-test_mode mode_name_list | all]
```

Data Types

```
chain_name_list  list
mode_name_list  list
```

ARGUMENTS

-view spec | existing_dft

Specifies the type of scan information to retrieve. Valid options are **spec** and **existing_dft**. Specifying the **spec** value shows only user-defined scan path information previously specified with the **set_scan_path** command. Specifying the **existing_dft** value shows only scan paths already inserted by the **insert_dft** command. The default behavior is to show both views.

-chain *chain_name_list* | all

When specified, retrieves and returns a list of scan cell names contained in the specified scan chains, or for all scan chains if **all** is specified. By default, the scan cells for all chains are returned.

-test_mode *mode_name_list* | all

Reports only on the specified test mode(s), or on all test modes if **all** is specified. For multiple modes, the lists are concatenated together. By default, the tool reports on the current test mode. If no test modes are defined, the default is to report on the `Internal_scan` mode.

DESCRIPTION

This command returns a Tcl list of scan cell names (as strings), according to the specified options. The information can include scan paths defined by the **set_scan_path** command, as well as scan paths inserted by the **insert_dft** command.

By default, the command returns the scan cell names for all chains, in both the **spec** and **existing_dft** views, in the current test mode.

EXAMPLES

The following example reports all the scan cells, of every scan chain, for every defined test mode:

```
prompt> set mode_list [all_test_modes]
      foreach p $mode_list {
        set chain_list [get_scan_chain_names -test_mode $p]
        foreach q $chain_list {
          set scan_cell_names_list [get_scan_cell_names -chain $q]
          foreach r $scan_cell_names_list {
            report_cell $r
          }
        }
      }
```

SEE ALSO

- current_test_mode(2)
- all_test_modes(2)
- get_scan_chain_names(2)
- report_dft_signal(2)
- report_scan_configuration(2)
- report_scan_path(2)
- set_dft_configuration(2)
- set_dft_signal(2)
- set_scan_configuration(2)
- set_scan_path(2)

get_scan_cells_of_chain

Returns a collection containing all scan cells of the specified scan chain.

SYNTAX

```
collection get_scan_cells_of_chain  
-chain chain_name  
[-test_mode test_mode]
```

Data Types

```
chain_name  string  
test_mode  string
```

ARGUMENTS

-chain *chain_name*

Specifies the name of the scan chain.

-test_mode *test_mode*

Specifies a mode name. The command returns only scan cells of this mode name in the Tcl collection. By default, this option is off.

DESCRIPTION

This command returns a Tcl collection containing all scan cells for the specified scan chain.

If CTL-modeled core scan segments exist in the scan chain,

- If the CTL-modeled core has its netlist in memory, the individual cells composing the scan segment(s) are returned. (However, remember that DFT configuration commands require CTL model scan segments to be referenced by their segment name, not by their individual scan cells.)
- If the CTL-modeled core has no netlist, the CTL-modeled cell itself is returned. This reference contains no information about the scan segment name.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example gets all scan cells of the *chain1* scan chain:

```
prompt> get_scan_cells_of_chain -chain "chain1"
```

SEE ALSO

`get_scan_chains(2)`

get_scan_chain_names

Returns a list containing the names of scan chains for the specified option filters.

SYNTAX

```
list get_scan_chain_names
    [-view spec | existing_dft]
    [-test_mode mode_name_list | all]
```

Data Types

mode_name_list list

ARGUMENTS

-view spec | existing_dft

Specifies the type of scan information to retrieve. Valid options are **spec** and **existing_dft**. Specifying the **spec** value shows only user-defined scan path information previously specified with the **set_scan_path** command. Specifying the **existing_dft** value shows only scan paths already inserted by the **insert_dft** command. The default behavior is to show both views.

-test_mode *mode_name_list* | all

Reports only on the specified test mode(s), or on all test modes if **all** is specified. For multiple modes, the lists are concatenated together. By default, the tool reports on the current test mode. If no test modes are defined, the default is to report on the Internal_scan mode.

DESCRIPTION

This command returns a Tcl list of scan chain names (as strings), according to the specified options. The information can include scan paths defined by the **set_scan_path** command, as well as scan paths inserted by the **insert_dft** command.

By default, the command returns the scan chain names for both **spec** and **existing_dft** views, in the current test mode.

EXAMPLES

The following example reports all the scan cells, of every scan chain, for every defined test mode:

```
prompt> set mode_list [all_test_modes]
foreach p $mode_list {
  set chain_list [get_scan_chain_names -test_mode $p]
  foreach q $chain_list {
    set scan_cell_names_list [get_scan_cell_names -chain $q]
    foreach r $scan_cell_names_list {
      report_cell $r
    }
  }
}
```

SEE ALSO

- current_test_mode(2)
- all_test_modes(2)
- get_scan_cell_names(2)
- report_dft_signal(2)
- report_scan_configuration(2)
- report_scan_path(2)
- set_dft_configuration(2)
- set_dft_signal(2)
- set_scan_configuration(2)
- set_scan_path(2)

get_scan_chains

Returns the number of scan chains in the current design.

SYNTAX

```
status get_scan_chains  
[-test_mode test_mode]
```

Data Types

test_mode string

ARGUMENTS

-test_mode *test_mode*

Specifies the target test mode to use to return the number of scan chains.

DESCRIPTION

This command returns the number of scan chains in the current design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example shows how to get the number of scan chains in the design:

```
prompt> get_scan_chains
```

SEE ALSO

```
get_scan_cells_of_chain(2)
```

get_scenarios

Returns a list of scenarios that match all of the specified criteria.

SYNTAX

```
list get_scenarios  
  [-setup true | false]  
  [-hold true | false]  
  [-leakage_power true | false]  
  [-dynamic_power true | false]  
  [-active true | false]  
  [pattern]  
  [-cts_mode true | false]  
  [-cts_corner min | max | min_max]
```

Data Types

pattern string

ARGUMENTS

-setup true | false

Compares against the setup option for each scenario.

-hold true | false

Compares against the hold option for each scenario.

-leakage_power true | false

Compares against the leakage power option for each scenario.

-dynamic_power true | false

Compares against the dynamic power option for each scenario.

-active true | false

Controls whether the command returns active or inactive scenarios. When this option is true, returns only scenarios that are currently active. When false, returns only scenarios that are currently inactive.

pattern

Matches scenario names against the specified pattern.

DESCRIPTION

This command returns a list of scenarios that match all the specified criteria.

If you do not specify any arguments, the command returns all scenarios in the current design, similar to the **all_scenarios** command.

If you specify only the *pattern* argument, the command returns a list of all scenarios whose name matches that pattern.

All arguments other than *pattern* and **-active** are tested against the options set by the **set_scenario_options** for each scenario. For more information about the scenario options, see the man page of the **set_scenario_options** command.

Multicorner-Multimode Support

This command uses information from both active and inactive scenarios.

EXAMPLES

The following example uses the **get_scenarios** command to activate a list of inactive scenarios that are configured to be used for setup and hold optimization.

```
prompt> set_active_scenarios \  
[get_scenarios -setup true -hold true -active false]  
Information: Scenario S4 is now active. (UID-1023)  
1
```

The following example uses the **get_scenarios** command to loop through and report on all CTS scenarios for clock tree reporting.

```
prompt> set_current [current_scenario]  
prompt> foreach scenario [get_scenarios -cts_mode true -active true] {  
  current_scenario $scenario  
  report_clock_tree  
}  
prompt> current_scenario $_current
```

SEE ALSO

report_scenario_options(2)
set_scenario_options(2)
all_scenarios(2)
current_scenario(2)
set_active_scenarios(2)

get_selection

Returns a collection that contains the objects in the current selection.

SYNTAX

```
collection get_selection
  [-slct_targets target_selection_bus
  [-slct_targets_operation operation]]
  [-create_slct_buses]
  [-name selection_bus]
  [-type object_type]
  [-design design]
  [-more_than more]
  [-fewer_than fewer]
  [-count]
  [-num max_objects]
  [-type_list]
```

Data Types

<i>target_selection_bus</i>	string
<i>operation</i>	string
<i>selection_bus</i>	string
<i>object_type</i>	string
<i>design</i>	string
<i>more</i>	integer
<i>fewer</i>	integer
<i>max_objects</i>	integer

ARGUMENTS

-slct_targets *target_selection_bus*

Specifies the name of the selection bus in which to store the result. When you use this option, the command returns the *target_selection_bus* value. By default, the command returns the result in a collection.

-slct_targets_operation *operation*

Specifies the operation used to store the result in the selection bus specified by the **-slct_targets** option. The valid values for the *operation* argument are **clear**, **add**, and **remove**.

You can use the **-slct_targets_operation** option only when you also use the **-slct_targets** option.

-create_slct_buses

Creates a new selection bus in which to store the result.

-name *selection_bus*

Specifies the selection bus from which the selection is taken.

-type *object_type*

Filters the selection by object type and returns only the objects of the specified *object_type*. The valid values for *object_type* and their descriptions are

```

design -- design object
port  -- port object
net   -- net object
cell  -- cell object
pin   -- pin object
path  -- timing path object

```

-design *design*

Returns only objects in the filter specified by *design*.

-more_than *more*

Returns 1 if the selection bus contains more than the number of objects specified by *more*; otherwise, returns 0.

This option and the **-type_list** option are mutually exclusive.

-fewer_than *fewer*

Returns 1 if the selection bus contains more than the number of objects specified by *fewer*; otherwise, returns 0.

This option and the **-type_list** option are mutually exclusive.

-count

Returns the number of elements contained in the selection bus.

This option and the **-type_list** option are mutually exclusive.

-num *max_objects*

Specifies the maximum number of objects in the collection. The tool returns *max_objects* objects or fewer depending on the number of objects in the current selection.

-type_list

Returns only the number of objects of each object type in the collection.

You cannot use this option with the **-more_than** option, the **-fewer_than** option, or the **-count** option.

DESCRIPTION

The **get_selection** command returns a collection that contains the current selection in the tool. By default, the command returns a collection handle (identifier) if any objects are selected. If no objects are selected, the command returns an empty string.

By default, the command returns a heterogeneous collection that contains all the objects that are currently selected. If you use the **-type** option, the command returns a homogeneous collection that has been filtered to contain only the objects of the *type* that you

specify.

You can enter the **get_selection** command at the command prompt, or you can nest it as an argument to another command, such as **query_objects**), or assign the **get_selection** result to a variable.

When you enter **get_selection** at the command prompt, it behaves as if you had called **query_objects** to display the objects in the collection. By default, **get_selection** displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

The "implicit query" property of **get_selection** allows you to display cells in a collection. However, if you want the flexibility provided by the options of the **query_objects** command (for example, to display the object class), use **get_selection** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns a collection containing the cell objects in the current selection:

```
prompt> get_selection -type cell
{I_PRGRM_CNT_TOP I_DATA_PATH I_REG_FILE I_STACK_TOP}
```

The following example assigns the collection to a variable named "collect_a" that is passed to the **query_objects** command:

```
prompt> set collect_a [get_selection -type cell]
{I_PRGRM_CNT_TOP I_DATA_PATH I_REG_FILE I_STACK_TOP}
prompt> query_objects $collect_a
{I_PRGRM_CNT_TOP I_DATA_PATH I_REG_FILE I_STACK_TOP}
```

SEE ALSO

change_selection(2)
collections(2)
filter_collection(2)
query_objects(2)

get_shift_register_chains

Creates a list of collections of shift-register cells in the current design, where each collection is an ordered set of shift-register chain cells.

SYNTAX

list of collection **get_shift_register_chains**
[-quiet]
[-power_domain ignore | skip | split]
[*patterns*]

Data Types

patterns list or collection

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-power_domain ignore | skip | split

Controls how the **get_shift_register_chains** command returns shift registers that cross power domains.

Specify **ignore** to ignore power domain information and return the shift registers even if they cross power-domain boundaries. This is the default.

Specify **skip** to skip (not return) shift registers that cross power-domain boundaries.

Specify **split** to return shift registers that cross power-domain boundaries as if they are separate shift registers, one shift register for each power domain. Note that this option does not modify the shift register logic; it only affects the results returned by this command.

patterns

Creates collections of shift-register chains whose names match the specified patterns. Patterns can include the asterisk (*) and question mark (?) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page.

You can also provide a collection of cells as a subset so that the command only returns shift-register chains from this subset of cells.

DESCRIPTION

The **get_shift_register_chains** command queries and returns shift registers identified during test-ready compile. Shift registers are identified by the **compile_ultra -scan** command when the **compile_seqmap_identify_shift_registers** variable is set to **true**.

The **get_shift_register_chains** command creates a list of collections of shift-register chains that match the specified criteria. By default, the command creates one collection for each shift-register chain found in the current design.

The order of shift-register elements in each collection is based on netlist connectivity: the first element is the shift register that is the head of the chain, the second element is the next shift register in the chain, and so on.

Note that if you have enabled synchronous shift-register identification by setting the **compile_seqmap_identify_shift_registers_with_synchronous_logic** variable to **true**, this command does not return synchronous-logic shift registers that have discrete combinational logic cells (other than buffers and inverters) between the register cells.

The shift-register elements within a collection can span different levels of design hierarchy.

If the command cannot find a shift-register chain that matches the criteria, the command issues a UID-921 warning message unless you specify the **-quiet** option.

You can use this command to assign the resulting list to a variable, then evaluate each collection of the list using **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries shift-register chains that contains shift registers with names that begin with "sr":

```
prompt> foreach chain [get_shift_register_chains sr* ] { query_objects $chain }
{sr_reg1 sr_reg2 sr_reg3 sr_reg4}
{sr_reg5 U23/shift_reg1 U23/shift_reg2}
{U4/shr22 U4/shr23 sr_reg6}
```

The following example queries shift-registers chains that match a collection of cells:

```
prompt> foreach chain [get_shift_register_chains [get_cells shr* ] { query_objects $chain }
{U4/shr22 U4/shr23 sr_reg6}
```

SEE ALSO

collections(2)
filter_collection(2)
query_objects(2)

```
collection_result_display_limit(3)  
wildcards(3)  
compile_seqmap_identify_shift_registers(3)  
compile_seqmap_identify_shift_registers_with_synchronous_logic(3)
```

get_site_rows

Returns a collection of site rows from the current design that match the specified criteria.

SYNTAX

```
collection get_site_rows
  [-quiet]
  [-regexp | -exact]
  [-nocase]
  [-filter expression]
  [-within region
   | -touching region
   | -intersect region]
  [patterns]
```

Data Types

```
expression  string
region      list of points
patterns    list
```

ARGUMENTS

-quiet

Suppresses warning and error messages, if no objects match. Does not suppress syntax error messages.

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each site row in the collection, the expression is evaluated based on the site row's attributes. If the expression evaluates to true, the site row is included in the result.

To see the list of site row attributes that you can use in the expression, use the **list_attributes -application -class site_row** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-within *region*

Creates a collection that contains all site rows that are completely inside the specified region and do not overlap the boundary. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, where each `{x y}` pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, and **-intersect** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the command searches for site rows in the entire design.

-touching *region*

Creates a collection that contains all site rows that are inside the specified region, including those that overlap the boundary. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, where each `{x y}` pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, and **-intersect** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the command searches for site rows in the entire design.

-intersect *region*

Creates a collection that contains all site rows that intersect the boundary of the specified region and at least part of the cell is outside of the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is `{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}`, where each `{x y}` pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, and **-intersect** options are mutually exclusive; you can specify only one. If you do not specify any of these

options, the command searches for site rows in the entire design.

patterns

Creates a collection of site rows whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

If you do not specify this argument, the command uses * (asterisk) as the default pattern.

DESCRIPTION

This command creates a collection of site rows by selecting site rows from the current design that match the specified criteria.

When using the **-within**, **-touching**, or **-intersect** option, note that the behavior differs depending on the format of the specified region. An object has both a boundary and a bounding box (bbox). In some cases, these attributes differ. When you specify a region in rectangle format, the tool searches objects by their bounding box. When you specify a region in polygon format, the tool searches objects by their boundary. Searching by bounding box is faster than searching by boundary.

The command returns a collection if any site rows match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example returns a collection that contains all the site rows in the current design.

```
prompt> get_site_rows
{SITE_ROW#12345 SITE_ROW#12346 SITE_ROW#12347 my_row1 my_row2}
```

The following example returns a collection that contains the site rows whose name is prefixed with my_row.

```
prompt> get_site_rows my_row*
{my_row1 my_row2}
```

The following example returns a collection that contains the site rows within the specified rectangle.

```
prompt> get_site_rows -within {{2 2} {250 150}}
{test_site_row1}
```

The following example returns a collection that contains the site rows that intersect the specified polygon.

```
prompt> get_site_rows \
-intersect {{5 5} {200 5} {200 100} {5 100} {5 5}}
{test_site_row1 test_site_row2}
```

SEE ALSO

collections(2)
filter_collection(2)
list_attributes(2)
query_objects(2)
create_site_row(2)
collection_result_display_limit(3)
wildcards(3)

get_supply_nets

Creates a collection of supply nets that match the specified criteria.

SYNTAX

```
collection get_supply_nets
  [-quiet]
  [-regexp]
  [-nocase]
  [-filter expression]
  [-hierarchical]
  [patterns]
```

Data Types

```
expression  string
patterns    list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the == and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each supply net in the collection, the expression is evaluated based on the supply net's attributes. If the expression evaluates to true, the supply net is included in the result.

To see the list of supply net attributes that you can use in the expression, use the **list_attributes -application -class supply_net** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-hierarchical

Searches for supply nets in the current scope and all of its descendant scopes.

If you specify this option, you can specify only simple names in the *patterns* argument; the name cannot contain hierarchy delimiter characters.

If you do not specify this option, the tool searches for supply nets only in the current scope. You can use hierarchy delimiter characters in the *patterns* argument to specify hierarchical supply nets relative to the current scope.

patterns

Creates a collection of supply nets whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

If you do not specify this option, the command uses * (asterisk) as the default pattern.

DESCRIPTION

This command creates a collection of supply nets by selecting supply nets from the current design that match the specified criteria.

If no supply nets match the criteria and the design is not linked, the tool automatically links the design.

The command returns a collection if any supply nets match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets all supply nets under the current scope:

```
prompt> get_supply_nets -hierarchical  
{VDD SUB_INST/VDD}
```

SEE ALSO

`create_supply_net(2)`
`report_supply_net(2)`
`collections(2)`
`filter_collection(2)`
`list_attributes(2)`
`query_objects(2)`
`win_select_objects(2)`
`collection_result_display_limit(3)`
`wildcards(3)`

get_supply_ports

Creates a collection of supply ports that match the specified criteria.

SYNTAX

```
collection get_supply_ports
  [-quiet]
  [-regex]
  [-nocase]
  [-filter expression]
  [-hierarchical]
  [patterns]
```

Data Types

```
expression  string
patterns    list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regex

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the == and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regex** command. When using the **-regex** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each supply port in the collection, the expression is evaluated based on the supply port's attributes. If the expression evaluates to true, the supply port is included in the result.

To see the list of supply port attributes that you can use in the expression, use the **list_attributes -application -class supply_port** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-hierarchical

Searches for supply ports in the current scope and all of its descendant scopes.

When you use the **-hierarchical** option, you can specify only simple names in the *patterns* option; the name cannot contain hierarchy delimiter characters.

By default, the tool searches for supply ports only in the current scope and you can use hierarchy delimiter characters in the *patterns* option to specify hierarchical supply ports relative to the current scope.

patterns

Creates a collection of supply ports whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

If you do not specify this option, the command uses * (asterisk) as the default pattern.

DESCRIPTION

This command creates a collection of supply ports by selecting supply ports from the current design that match the specified criteria.

If no supply ports match the criteria and the design is not linked, the tool automatically links the design.

The command returns a collection if any supply ports match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets all supply ports under the current scope:

```
prompt> get_supply_ports -hierarchical  
{VN SUB_INST/VN12}
```

SEE ALSO

`create_supply_port(2)`
`report_supply_port(2)`
`collections(2)`
`filter_collection(2)`
`list_attributes(2)`
`query_objects(2)`
`collection_result_display_limit(3)`
`wildcards(3)`

get_switching_activity

Gets switching activity information on nets, pins, ports and cells in the current design.

SYNTAX

```
string get_switching_activity  
  [-state_condition state_condition]  
  [-path_sources path_sources]  
  [-rise]  
  [-fall]  
  [-related_clock]  
  [-scenarios scenario_list]  
  object_list
```

Data Types

```
state_condition  string  
path_sources    list  
scenario_list   list  
object_list     list
```

ARGUMENTS

-state_condition *state_condition*

Specifies the state condition for reporting state-dependent activities. State dependent activities can be reported when the object is already annotated with state-dependent activities. The state condition specified with this argument must match to a state condition already annotated for this object. Use -state_condition default to specify the default state condition.

-path_sources *path_sources*

Specifies the path sources when reporting path-dependent activities. This can be used when the object is already annotated with path-dependent activities. The path sources specified with this argument must be the same as those annotated for this object. If -path_sources option is used to specify the path_sources value, then -state_condition option must be specified to specify the value of the path_sources.

-rise

This option is used with the **-state_condition** option. It reports rise transition.

-fall

This option is used with the **-state_condition** option. It reports fall transition.

-related_clock

This option is used to print the clock domain of the objects.

-scenarios *scenario_list*

Specifies the scenarios to be considered. If this option is not specified, then only the current scenario is considered. Output of the command is generated separately for each scenario.

object_list

Specifies a net, pin, port or cell in the current design for which the values of static probability, the toggle rate and the type of activity are to be reported.

DESCRIPTION

This command is used to report switching activity information on nets, ports, pins and cells. Switching activity information includes toggle rate, static probability and activity type.

Note that *object_list* argument should always be provided to the command. The activity type reported by the command can only be one of 'annotated', 'simulated', 'inferred', 'propagated', or 'default'.

For more statistics on the switching activity annotation on the current design, use the **report_activity** command.

Multicorner-Multimode Support

This command uses information from the current scenario only if **-scenarios** option is not specified.

EXAMPLES

The following **get_switching_activity** command reports the toggle rate and static probability values for a list of objects for the current scenario in the design.

```
prompt> get_switching_activity {U1/A U1/B}
(U1/A: probability = 0.800000, toggle_rate = 0.900000, type = annotated)
(U1/B: probability = 0.800000, toggle_rate = 0.900000, type = annotated)
```

The following **get_switching_activity** command reports the activity information for a list of objects for the current scenario with related_clock information.

```
prompt> get_switching_activity {U1/A U1/B} -related_clock
(U1/A: probability = 0.800000, toggle_rate = 0.900000, type = annotated, related_clock = clk)
(U1/B: probability = 0.800000, toggle_rate = 0.900000, type = annotated, related_clock = clk)
```

SEE ALSO

read_saif(2)
report_power(2)
reset_switching_activity(2)
set_switching_activity(2)

get_terminals

Creates a collection of terminals that match the specified criteria.

SYNTAX

```
collection get_terminals
  [-quiet]
  [-regex]
  [-nocase]
  [-filter expression]
  [patterns | -of_objects port_list]
```

Data Types

```
expression  string
patterns    list
port_list   collection
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regex

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the == and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regex** command. When using the **-regex** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each terminal in the collection, the expression is evaluated based on the terminal's attributes. If the expression evaluates to true, the terminal is included in the result.

To see the list of terminal attributes that you can use in the expression, use the **list_attributes -application -class terminal** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

patterns

Creates a collection of terminals whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify either of these arguments, the command uses * (asterisk) as the default pattern.

-of_objects port_list

Creates a collection of terminals connected to the specified ports.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify either of these arguments, the command uses * (asterisk) as the default pattern.

DESCRIPTION

This command creates a collection of terminals by selecting terminals from the current design that match the specified criteria.

The command returns a collection if any terminals match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a collection of terminals named reset or with a prefix of clock:

```
prompt> get_terminals [list reset clock*]  
{reset clock}
```

SEE ALSO

create_terminal(2)
remove_terminal(2)
collections(2)
filter_collection(2)
list_attributes(2)
query_objects(2)
collection_result_display_limit(3)
wildcards(3)

get_timing_paths

Creates a collection of timing paths for custom reporting and other processing.

SYNTAX

```
collection get_timing_paths
  [-to to_list
  | -rise_to rise_to_list
  | -fall_to fall_to_list]
  [-from from_list
  | -rise_from rise_from_list
  | -fall_from fall_from_list]
  [-through through_list
  | -rise_through rise_through_list
  | -fall_through fall_through_list]
  [-exclude exclude_list
  | -rise_exclude rise_exclude_list
  | -fall_exclude fall_exclude_list]
  [-delay_type delay_type]
  [-nworst paths_per_endpoint]
  [-max_paths max_path_count]
  [-enable_preset_clear_arcs]
  [-group group_name]
  [-greater greater_limit]
  [-lesser lesser_limit]
  [-slack_greater_than greater_slack_limit]
  [-slack_lesser_than lesser_slack_limit]
  [-include_hierarchical_pins]
  [-path_type full_clock_expanded | full]
  [-unique_pins]
  [-start_end_pair]
  [-scenarios scenario_list]
```

Data Types

<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list
<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list
<i>exclude_list</i>	list
<i>rise_exclude_list</i>	list
<i>fall_exclude_list</i>	list

```

delay_type      string
paths_per_endpoint integer
max_path_count  integer
group_name      list
greater_limit   float
lesser_limit    float
greater_slack_limit float
lesser_slack_limit float
scenario_list   list

```

ARGUMENTS

-to *to_list*

Specifies the to pins, ports, nets, or clocks. Path endpoints are the output ports or data pins of registers. If you specify a clock, the command considers all endpoints that are constrained by the clock.

-rise_to *rise_to_list*

This option is similar to the **-to** option, but applies only to paths rising at the endpoint. If you specify a clock, this option selects the paths that are captured by the rising edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-fall_to *fall_to_list*

This option is similar to the **-to** option, but applies only to paths falling at the endpoint. If you specify a clock, this option selects the paths that are captured by the falling edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-from *from_list*

Specifies the from pins, ports, nets, or clocks. Path startpoints are the input ports or clock pins of registers. If you specify a clock, the command considers all startpoints clocked by the clock.

-rise_from *rise_from_list*

This option is similar to the **-from** option, but applies only to paths rising from the specified objects. If you specify a clock, this option selects the startpoints whose paths are launched by the rising edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-fall_from *fall_from_list*

This option is similar to the **-from** option, but applies only to paths falling from the specified objects. If you specify a clock, this option selects the startpoints whose paths are launched by the falling edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-through *through_list*

Reports only paths that pass through the named pins, ports, or clocks. If you do not use the **-through** option, the command defaults to reporting the longest path to an output port if the design has no timing constraints. If it has timing constraints, the default is to report the path with the worst slack within each path group if you do not use the **-group** option. If you use **-group**, the default is to report the path with the worst slack within the group specified by the *group_name* value.

If you specify the **-through** option only once, the tool reports only the paths that travel through one or more of the objects in the list. You can use **-through** more than once in one command invocation. For a discussion of the use of multiple **-through** options, see the *DESCRIPTION* section.

-rise_through *rise_through_list*

This option is similar to the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than once in a single command invocation. For a discussion of multiple **-through**, **-rise_through**, and **-fall_through** options, see the DESCRIPTION section.

-fall_through *fall_through_list*

This option is similar to the **-through** option, but applies only to paths with a falling transition at the specified objects. You can specify **-fall_through** more than once in a single command invocation. For a discussion of multiple **-through**, **-rise_through**, and **-fall_through** options, see the DESCRIPTION section.

-exclude *exclude_list*

Excludes all data paths from/through/to the named pins, ports, nets, and cell instances. Any data path that starts from, contains, or ends on a listed object is excluded. If you specify a cell instance, data paths that include any pin of that cell instance are excluded. This option has higher precedence than the **-from**, **-through**, **-to**, and similar options.

The exclusion does not apply to clock paths, even when the **-path_type full** or **-path_type full_clock_expanded** option is used. It does not apply to borrowing paths from the **-trace_latch_borrow** option.

This option is not applied to clock pins.

-rise_exclude *rise_exclude_list*

Same as the **-exclude** option, but applies only to paths rising at the named pins, ports, nets, and cell instances.

-fall_exclude *fall_exclude_list*

Same as the **-exclude** option, but applies only to paths falling at the named pins, ports, nets, and cell instances.

-delay_type *delay_type*

Specifies the type of path delay. The valid values are **max**, **min**, **min_max**, **max_rise**, **max_fall**, **min_rise**, and **min_fall**. The rise or fall in the *delay_type* refers to a rising or falling transition at the path endpoint.

-nworst *paths_per_endpoint*

Specifies the maximum number of paths to get per endpoint. The default is 1, which gets only the single worst path ending at a given endpoint.

-max_paths *max_path_count*

Specifies the number of paths to get per path group by default, or the number of paths to get from the whole design if the **timing_report_fast_mode** variable is set to true. The default is 1.

-enable_preset_clear_arcs

Enables asynchronous preset and clear arcs for the timing path collection. By default, asynchronous timing arcs are disabled during timing verification.

-group *group_name*

Restricts the collection to paths in this *group_name*. Use the **group_path** or **create_clock** command to group paths.

-greater *greater_limit*

Selects only those paths that have a delay greater than the *greater_limit* value. The **-greater** and **-lesser** options can be combined to select only those paths inside or outside of a given delay range. The specified value is in main library units.

-lesser *lesser_limit*

Selects only those paths that have a delay less than the *lesser_limit* value. The **-greater** and **-lesser** options can be combined to select only those paths inside or outside of a given delay range. The specified value is in main library units.

-slack_greater_than greater_slack_limit

Selects only those paths with a slack greater than *greater_slack_limit*. The **-slack_greater_than** option can be combined with the **-slack_lesser_than** option to select only those paths inside or outside of a given slack range. The specified value is in main library units.

-slack_lesser_than lesser_slack_limit

Selects only those paths with a slack less than *lesser_slack_limit*. The **-slack_greater_than** option can be combined with the **-slack_lesser_than** option to select only those paths inside or outside of a given slack range. The specified value is in main library units.

-include_hierarchical_pins

Specifies for the returned timing paths to contain points for each hierarchical pin crossed, as well as for all leaf pins in the path. It also sets to **true** their **hierarchical** attributes. Note that clock paths are always hierarchical.

-path_type full_clock_expanded | full

Specifies whether or not to calculate launch and capture clock paths (if they exist) for the selected paths. The valid values are **full_clock_expanded** and **full**. When you specify *full_clock_expanded*, the tool calculates clock paths. When you specify **full**, it does not calculate them. You can access the clock path data of a path (if calculated) via its **launch_clock_paths** and **capture_clock_paths** attributes.

-unique_pins

Causes only the single worst timing path through any given sequence of pins to be reported. No other paths are reported for the same sequence of pins from startpoint to endpoint. For example, if the worst path starts with a rising edge at the first pin of a pin sequence, then paths starting with a falling edge are not reported for that sequence. For non-unate logic such as XOR gates, this greatly reduces the number of paths reported because of the large number of possible rising/falling edge combinations through the sequence of pins. Using this option can require longer runtimes when used with the **-nworst n** option because many paths must be analyzed to find the worst path through each pin sequence, but only the worst path is reported and counted toward the total number of requested paths.

-start_end_pair

Limits the collection of paths returned to the single worst timing path per each combination of startpoint and endpoint found. No other paths are included that have the same startpoint and same endpoint. For example, if the worst path starts at a register output pin ff1/Q and ends at a register input pin ff5/D, the collection will omit all other less critical paths from ff1/Q to ff5/D. Using this option can require longer runtimes when used with the **-nworst n** option because many paths must be analyzed to find the worst path through each pin pair, but only the worst path in each case is selected and counted toward the total number of requested paths.

-scenarios scenario_list

Gets timing paths from the specified scenarios. Inactive scenarios are skipped.

If you do not specify this option, only the timing paths from current scenario are returned.

DESCRIPTION

This command creates a collection of paths for custom reporting or other operations. You can use the **foreach_in_collection** command to iterate among the paths in the collection. You can use the **get_attribute** and **collection** commands to obtain

information about the paths. The following attributes are supported on timing paths:

- capture_clock_paths
- clock_uncertainty
- clock_path
- crpr_common_point
- crpr_value
- endpoint
- endpoint_clock
- endpoint_clock_close_edge_type
- endpoint_clock_close_edge_value
- endpoint_clock_is_inverted
- endpoint_clock_is_propagated
- endpoint_clock_latency
- endpoint_clock_open_edge_type
- endpoint_clock_open_edge_value
- endpoint_clock_pin
- endpoint_hold_time_value
- endpoint_is_level_sensitive
- endpoint_output_delay_value
- endpoint_recovery_time_value
- endpoint_removal_time_value
- endpoint_setup_time_value
- hierarchical
- launch_clock_paths
- object_class
- path_group
- path_type
- points
- scenario
- slack
- startpoint
- startpoint_clock
- startpoint_clock_is_inverted
- startpoint_clock_is_propagated
- startpoint_clock_latency
- startpoint_clock_open_edge_type
- startpoint_clock_open_edge_value
- startpoint_input_delay_value
- startpoint_is_level_sensitive
- time_borrowed_from_endpoint
- time_lent_to_startpoint

One attribute of a timing path is the **points** collection. A point corresponds to a pin or port along the path. You can iterate through these points by using the **foreach_in_collection** command and get their attributes by using the **get_attribute** command. The following attributes are available for points of a timing path:

- arrival
- object
- object_class
- rise_fall
- slack

See the **collections** and **foreach_in_collection** man pages for detailed information.

By default, the **get_timing_paths** command uses a reporting engine that can run extremely fast on large designs, especially for larger values of the **nworst** and **max_paths** arguments. For designs that have multiple timing paths with identical slack, this engine

might report different paths (with the same slack value) than a previous engine that was once in use.

You can use multiple iterations of the **-through**, **-rise_through**, and **-fall_through** options in a single command to specify paths that traverse multiple points in the design. The following example specifies paths beginning at A1, passing through B1, then through C1, and ending at D1.

```
prompt> get_timing_paths -from A1\
        -through B1 -through C1 -to D1
```

If more than one object is specified by one **-through** option, the path can pass through any of the objects. The following example specifies paths beginning at A1, passing through either B1 or B2, then passing through either C1 or C2, and ending at D1.

```
prompt> get_timing_paths -from A1 -through {B1 B2} -through {C1 C2} -to D1
```

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following example prints out the startpoint name, endpoint name, and slack of the worst path in each path group.

```
proc custom_report_worst_path_per_group {} {
  echo [format "%-20s %-20s %7s" "From" "To" "Slack"]
  echo "-----"
  foreach_in_collection path [get_timing_paths] {
    set slack [get_attribute $path slack]
    set startpoint [get_attribute $path startpoint]
    set endpoint [get_attribute $path endpoint]
    echo [format "%-20s %-20s %s" [get_attribute $startpoint full_name] \
      [get_attribute $endpoint full_name] $slack]
  }
}
```

```
prompt> custom_report_worst_path_per_group
```

From	To	Slack
ffa/CP	QA	0.1977
ffb/CP	ffd/D	3.8834

The following example shows total negative slack, total positive slack, and worst negative slack for the current design.

```
proc report_design_slack_information {} {
  set design_tns 0
  set design_wns 100000
  set design_tps 0
  foreach_in_collection group [get_path_groups *] {
    set group_tns 0
    set group_wns 100000
    set group_tps 0
    foreach_in_collection path [get_timing_paths -nworst 10000 -group [get_object_name $group]]{
      set slack [get_attribute $path slack]
```

```

    if {$slack < $group_wns} {
        set group_wns $slack
        if {$slack < $design_wns} {
            set design_wns $slack
        }
    }
    if {$slack < 0.0} {
        set group_tns [expr $group_tns + $slack]
    } else {
        set group_tps [expr $group_tps + $slack]
    }
}
set design_tns [expr $design_tns + $group_tns]
set design_tps [expr $design_tps + $group_tps]
set group_name [get_attribute $group full_name]
echo [format "Group '%s' Worst Negative Slack : %g" $group_name $group_wns]
echo [format "Group '%s' Total Negative Slack : %g" $group_name $group_tns]
echo [format "Group '%s' Total Positive Slack : %g" $group_name $group_tps]
echo ""
}
echo "-----"
echo [format "Design Worst Negative Slack : %g" $design_wns]
echo [format "Design Total Negative Slack : %g" $design_tns]
echo [format "Design Total Positive Slack : %g" $design_tps]
}

```

prompt> **report_design_slack_information**

Group 'CLK' Worst Negative Slack : -3.1166
 Group 'CLK' Total Negative Slack : -232.986
 Group 'CLK' Total Positive Slack : 4.5656

Group 'vclk' Worst Negative Slack : -4.0213
 Group 'vclk' Total Negative Slack : -46.1982
 Group 'vclk' Total Positive Slack : 0

 Design Worst Negative Slack : -4.0213
 Design Total Negative Slack : -279.184
 Design Total Positive Slack : 4.5656

SEE ALSO

collections(2)
 create_clock(2)
 foreach_in_collection(2)
 get_attribute(2)
 group_path(2)
 report_timing(2)
 timing_report_fast_mode(3)

get_tracks

Creates a collection of track objects that match the specified criteria.

SYNTAX

```
collection get_tracks
  [-quiet]
  [-filter expression]
  [-within rectangle
    | -touching rectangle
    | -intersect rectangle
    | -at at_point]
  [patterns | -of_objects layers]
```

Data Types

```
expression  string
rectangle   list of points
at_point    point
patterns    string
layers      collection
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-filter *expression*

Filters the collection with the specified expression. For each track in the collection, the expression is evaluated based on the track's attributes. If the expression evaluates to true, the track is included in the result.

You can use the following track attributes in the expression:

Attribute	Description
bbox	Coordinates of the bounding box of the track object
name	Name of the track object
layer	Name of the layer that the track is on
direction	Direction of the wire tracks, vertical or horizontal
count	Number of parallel wire tracks in the track object
space	Pitch of the wire tracks in the track object
start	Coordinates of the lower-left corner of the object
stop	Coordinates of the upper-right corner of the object

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-within *rectangle*

Creates a collection that contains all tracks that are completely inside the specified rectangle and do not overlap the boundary.

The format for specifying the rectangle is `{{llx lly} {urx ury}}` or `{llx lly urx ury}`, which specifies the lower-left and upper-right corners of the rectangle.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, **-intersect**, and **-at** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the command searches the entire design.

-touching *rectangle*

Creates a collection that contains all tracks that are inside the specified rectangle, including those that overlap the boundary.

The format for specifying the rectangle is the same as for the **-within** argument.

-intersect *rectangle*

Creates a collection that contains all tracks that intersect the boundary of the specified rectangle and at least part of the cell is outside of the specified rectangle.

The format for specifying the rectangle is the same as for the **-within** argument.

-at *at_point*

Creates a collection that contains all tracks at the specified point. The format for specifying a point is `{x y}`.

patterns

Creates a collection of tracks whose names match the specified patterns. Track names use the naming convention `TRACK_n`, where *n* is an integer. Patterns can include the `*` (asterisk) and `?` (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case-sensitive.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify either of these arguments, the command uses `*` (asterisk) as the default pattern.

-of_objects *layers*

Creates a collection of tracks that are on the specified layers.

DESCRIPTION

This command creates a collection of track objects by selecting the track objects in the current design that match specified criteria.

When using the **-within**, **-touching**, or **-intersect** option, note that the behavior differs depending on the format of the specified region. An object has both a boundary and a bounding box (bbox). In some cases, these attributes differ. When you specify a region in rectangle format, the tool searches objects by their bounding box. When you specify a region in polygon format, the tool searches objects by their boundary. Searching by bounding box is faster than searching by boundary.

The command returns a collection if any tracks match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example returns the tracks within the rectangle with the lower-left corner at {2 2} and the upper-right corner at {25 25}.

```
prompt> get_tracks * -within {{2 2} {25 25}}
{TRACK_5389}
```

The following example returns the tracks that are located on the METAL2 layer.

```
prompt> get_tracks -filter "layer~=METAL2"
{TRACK_4683}
```

The following example returns all tracks.

```
prompt> get_tracks "TRACK_*"
{TRACK_4683 TRACK_5389}
```

The following example shows how to use the **get_attribute** command to query the value of a track attribute.

```
prompt> get_attribute [get_tracks TRACK_4683] layer
METAL2
prompt> get_attribute [get_tracks TRACK_4683] bbox
{{0.000 0.000} {100.000 100.000}}
prompt> get_attribute [get_tracks TRACK_4683] start
0.000 0.000
```

To get a complete report of the attribute values of all tracks, use the **report_attribute** command, as shown in the following example.

```
prompt> report_attribute -application [get_tracks TRACK_4683]
Design Object   Type   Attribute Name  Value
-----
CORE TRACK_4683 string  bbox           {0.000 0.000} {100.000 100.000}
CORE TRACK_4683 int     cell_id        3
CORE TRACK_4683 string  layer          METAL2
CORE TRACK_4683 string  name           TRACK_4683
CORE TRACK_4683 string  object_class   track
CORE TRACK_4683 int     object_id      4683
CORE TRACK_4683 string  direction     horizontal
CORE TRACK_4683 string  start         0.000 0.0000
CORE TRACK_4683 string  stop          100.000 100.000
CORE TRACK_4683 int     count         51
CORE TRACK_4683 double  space         2.000
```

SEE ALSO

create_track(2)

remove_track(2)
get_attribute(2)
list_attributes(2)
report_attribute(2)
collections(2)
filter_collection(2)
query_objects(2)
collection_result_display_limit(3)
wildcards(3)

get_via_rules

Returns a collection of via_rules from current library.

SYNTAX

```
collection get_via_rules
  [-filter expression]
  [-quiet]
  [-regex]
  [-nocase]
  [-exact]
  [-expect exact_count]
  [-expect_at_least minimum_count]
  [-expect_each_pattern_matches]
  [patterns]
```

Data Types

```
expression    string
exact_count   int
minimum_count int
patterns      string
```

ARGUMENTS

-filter *expression*

Filters the collection with *expression*. For any via rule that match the patterns option, the expression is evaluated based on the via rule's attributes. If the expression evaluates to *true*, the via rule is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regex

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regex** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

patterns

Matches via rule names against the *patterns* argument. The *patterns* argument can include the wildcard character "*".

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses * (asterisk) as the pattern.

DESCRIPTION

The **get_via_rules** command creates a collection of via rules from the current library that match certain criteria. The command returns a collection of via rules if any via rule matches the criteria. If no objects match the criteria, the empty string is returned.

You can use the **get_via_rules** command at the command prompt, or you can nest it as an argument to another command, such as **report_attributes**. In addition, you can assign the **get_via_rules** result to a variable.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example returns a collection that contains all the via rules in the current library.

```
prompt> get_via_rules
{VR1 VL1}
```

SEE ALSO

collections(2)
query_objects(2)
collection_result_display_limit(3)

get_zero_interconnect_delay_mode

Reports whether or not the timer is currently using zero interconnect delay mode.

SYNTAX

status **get_zero_interconnect_delay_mode**

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command checks the timer to determine if it is currently using zero interconnect delay mode. The command returns 1 if the timer is using zero interconnect delay mode; otherwise it returns 0.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

set_zero_interconnect_delay_mode(2)

getenv

Returns the value of a system environment variable.

SYNTAX

```
string getenv  
    variable_name
```

Data Types

```
variable_name  string
```

ARGUMENTS

variable_name

Specifies the name of the environment variable to be retrieved.

DESCRIPTION

The **getenv** command searches the system environment for the specified *variable_name* and sets the result of the command to the value of the environment variable. If the variable is not defined in the environment, the command returns a Tcl error. The command is catchable.

Environment variables are stored in the **env** Tcl array variable. The **getenv**, **setenv**, and **printenv** environment commands are convenience functions to interact with this array.

The application you are running inherited the initial values for environment variables from its parent process (that is, the shell from which you invoked the application). If you set the variable to a new value using the **setenv** command, you see the new value within the application and within any new child processes you initiate from the application using the **exec** command. However, these new values are not exported to the parent process. Further, if you set an environment variable using the appropriate system command in a shell you invoke using the **exec** command, that value is not reflected in the current application.

See the **set**, **unset**, and **printvar** commands for information about working with non-environment variables.

EXAMPLES

In the following example, **getenv** returns you to your home directory:


```
prompt> set home [getenv "HOME"]  
/users/disk1/bill
```

```
prompt> cd $home
```

```
prompt> pwd  
/users/disk1/bill
```

In the following example, **setenv** changes the value of an environment variable:

```
prompt> getenv PRINTER  
laser1
```

```
prompt> setenv PRINTER "laser3"  
laser3
```

```
prompt> getenv PRINTER  
laser3
```

In the following example, the requested environment variable is not defined. The error message shows that the Tcl variable **env** was indexed with the value UNDEFINED, which resulted in an error. In the second command, **catch** is used to suppress the message.

```
prompt> getenv "UNDEFINED"  
Error: can't read "env(UNDEFINED)": no such element in array  
Use error_info for more info. (CMD-013)
```

```
prompt> if {[catch {getenv "UNDEFINED"} msg]} {  
    setenv UNDEFINED 1  
}
```

SEE ALSO

- catch(2)
- exec(2)
- printenv(2)
- printvar(2)
- set(2)
- setenv(2)
- unsetenv(2)
- unset(2)

group

Creates a new level of hierarchy.

SYNTAX

```
status group
  [cell_list | -logic | -pla | -fsm]
  [-soft
   | -hdl_block block_name
   | -hdl_all_blocks
   | -hdl_bussed]
  [-design_name design_name]
  [-cell_name cell_name]
  [-except exclude_list]
```

Data Types

```
cell_list    list
block_name   string
design_name  string
cell_name    string
exclude_list list
```

ARGUMENTS

cell_list

Groups a list of cells in the current design into a new level of hierarchy. If more than one cell is specified, they must be enclosed in braces ({}). Cell names cannot be specified with the following options:

- logic
- pla
- fsm
- hdl_block
- hdl_all_blocks
- hdl_bussed

Note that control logic cells (those designated as type "c" in the report displayed by using the **report_cell** command) must be grouped with the cells they are controlling.

-logic

Groups combinational cells into a new level of hierarchy. A cell is combinational if it is a leaf or library cell and its output pins have Boolean functions specified. The **-logic** option cannot be used with the *cell_list* argument or with the **-pla** or **-fsm** options.

-pla

Groups programmable logic array (PLA) specifications in the design into a level of hierarchy. The **-pla** option cannot be used with the **-logic** or **-fsm** options or with the *cell_list* argument. The **-pla** option is used only when the design contains blocks read as a PLA. To group combinational gates in the PLA output, use the **group** command with the **-logic -design_name design_name** options. After running the **compile** command, the design can be written out as a PLA.

-fsm

Groups cells that are part of a finite state machine design into a new level of hierarchy suitable for finite state machine extraction. Before using the **group** command, use the **set_fsm_state_vector** command to specify the sequential elements used to store the state of the finite state machine.

Cells in the transitive fanin or fanout of these state vector cells are grouped into the new hierarchy level. Noncombinational cells in the transitive fanin or fanout of the state vector cells are not included in the new hierarchy level. Upon completion of the group, the new design can be extracted into a state table format. This option cannot be used with the *cell_list* argument or the **-logic** or **-pla** options.

-soft

Specifies that cells be part of the same placement group. When you use this option, the **group** command sets the **group_name** attribute to the selected cells only and the design hierarchy is not modified. Use the **-soft** option with the **-logic**, **-pla**, and **-fsm** options. You cannot use the **-soft** option with the **-hdl_block**, **-hdl_all_blocks**, or **-hdl_bussed** options. You can set the value of the **group_name** attribute by using the **-cell_name** option.

-hdl_block block_name

Groups the logic cells for a labeled HDL block in the RTL file. The **-hdl_block** option requires that you specify the label name of an HDL block in the current instance. Nested HDL block labels are separated by a forward slash (/); for example, *my_process/my_function*.

block_name must reference a single label name. Wildcards are supported for label names, but they do not match hierarchy characters. If wildcards match multiple labels, only the first one is used.

For details on grouping HDL blocks, see the DESCRIPTION section.

-hdl_all_blocks

Groups all HDL blocks into a level of hierarchy. If you use the **-hdl_all_blocks** option alone, it recursively groups all HDL blocks in the current instance down. If you use it with the **-hdl_block** option, all HDL blocks from the *block_name* label down are recursively grouped.

For details on grouping HDL blocks, see the DESCRIPTION section.

-hdl_bussed

Places each group of bused gates created from the HDL into a level of hierarchy. If you use it with the **-hdl_block** option, the HDL blocks specified by the *block_name* argument are recursively grouped.

For details on grouping HDL blocks, see the DESCRIPTION section.

-design_name design_name

Names the design containing the new level of hierarchy. This name cannot already exist in the current design. Do not use the **-design_name** option when you use the **-hdl_all_blocks** or **-hdl_bussed** options.

-cell_name cell_name

Names the instance of the new design. By default, the command generates a unique cell name. Do not use the **-cell_name** option when you use the **-hdl_all_blocks** or **-hdl_bussed** options. Unique cell names are generated based on the **unique_cell_prefix** attribute in the current design. You can use the **get_attribute** command to obtain the value of the

unique_cell_prefix attribute. You can set this value by using the **set_attribute** command. Note that there is no variable controlling the creation of unique cell names.

-except *exclude_list*

Specifies a list of cells in the current design to exclude from grouping. If more than one cell is specified, they must be enclosed in braces ({}). Do not use the **-except** option with the **-hdl_block**, **-hdl_all_blocks**, or **-hdl_bussed** options.

DESCRIPTION

This command groups any number of cells or instances in the current design into a new design, creating a new level of hierarchy. The new design name must be defined by using the **-design_name** option (unless you are using the **-hdl_all_blocks** option or the **-hdl_bussed** option). The new design is copied into the current design.

To specify the cells to group into a new design, use one of the following options:

```
cell_list
-logic
-pla
-fsm
-hdl_block
-hdl_all_blocks
-hdl_bussed
```

Specify only one of these options (except for the **-hdl_all_blocks** and **-hdl_bussed** options, which can be used together or with the **-hdl_block** option). The *cell_list* argument can contain cells from anywhere in the hierarchy, but the parent of the cells in the list must be the same.

Use the **-cell_name** option to specify the instance name of the new design. Otherwise, the following naming format is used where *integer* is an integer value that ensures a unique name:

```
cell{integer}
```

The hierarchy below a grouped block can also be grouped by issuing the **group** command specifying the new cell names.

Names for each level of hierarchy are automatically generated from the label of the grouped block. The names have the following format where *label* represents the label assigned to the block and *digits* represents an integer that makes the name unique.

HDL_BLOCK is the default label.

```
label_digits
```

Ports in the new design are named the same as the nets to which they are connected in the current design. The direction of each port in the new design is determined for pins on the net according to the following table:

Inside pins	Outside pins	Resulting port direction
-----	-----	-----
IN	OUT	IN
OUT	IN	OUT
IN & OUT	OUT	IN
IN & OUT	IN	OUT
IN	IN & OUT	IN
OUT	IN & OUT	OUT
IN & OUT	IN & OUT	INOUT (bidir)

You can use the **-soft** option to define cell grouping constraints, which can be used by the layout placement tools and can be written to a file by using the **write_constraints** command. The **group -soft** command specifies that cells be part of the same placement

group. With this option, the **group** command sets only the **group_name** attribute for the cells in *cell_list* and the design hierarchy is not modified. You can set the value of the **group_name** attribute by using the **-cell_name** option. The **write_constraints** command reports cells with the *group_name* attribute as grouping constraints for placement tools. To remove the **group_name** attribute, use the **ungroup -soft** command or the **remove_attribute** command.

For the **-hdl_block**, **-hdl_all_blocks**, and **-hdl_bussed** options, only HDL blocks within the current instance can be referenced. To reference HDL blocks inside submodules, use the **current_instance** command to change the current instance to the parent block containing the HDL block. References to recursive grouping in the ARGUMENTS section refer to nested labels within the current instance, not to logical design hierarchy.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information. However, in order to preserve constraints and attributes for inactive scenarios, the tool automatically activates inactive scenarios before the hierarchy is changed and deactivates them before the command returns. Due to the overhead, if there are many **group** commands in the script, it is more runtime efficient to activate inactive scenarios once before the **group** command executions and deactivate them after all the **group** command executions finish.

EXAMPLES

The following example groups two cells into a new design:

```
prompt> group {u1 u2} -design_name NEW \
        -cell_name U
```

The following example groups cells with names beginning with *ANALOG* into a design:

```
prompt> group \
        -design_name ANALOG_CELLS [get_cells ANALOG*]
```

The following example groups all cells in the HDL function named *bar* and the process named *proc* into a design:

```
prompt> group -hdl_block proc/bar \
        -design_name my_hdl_block
```

The following example sets the current design to the new design and then uses the **group** command to group a nested function named *my_other_function* in the new design:

```
prompt> current_design my_hdl_block

prompt> group -hdl_block my_other_function
```

The following example groups all bused gates under the process named *proc* into separate levels of hierarchy (or designs):

```
prompt> group -hdl_block proc -hdl_bussed
```

The following example sets the *group_name* attribute to the name *proc* for all combinational cells. The design hierarchy is not modified.

```
prompt> group -logic -soft -cell_name proc
```

The following example creates a new hierarchy under a design named *BOT* with cells named *A*, *B*, and *C*. The *PROC/U1* is an instance of *BOT*.

```
prompt> group \
        -design_name new_design {PROC/U1/A PROC/U1/B PROC/U1/C}
```

SEE ALSO

change_names(2)
link(2)
set_fsm_state_vector(2)
ungroup(2)
set_active_scenarios(2)

group_path

Groups a set of paths for cost function calculations.

SYNTAX

```
status group_path
[-weight weight_value]
[-critical_range range_value]
[-default | -name group_name]
[-from from_list
 | -rise_from rise_from_list
 | -fall_from fall_from_list]
[-through through_list
 | -rise_through rise_through_list
 | -fall_through fall_through_list]
[-to to_list
 | -rise_to rise_to_list
 | -fall_to fall_to_list]
[-comment comment_string]
[-priority priority_level]
```

Data Types

<i>weight_value</i>	float
<i>range_value</i>	float
<i>group_name</i>	string
<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list
<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list
<i>comment_string</i>	string
<i>priority_level</i>	integer

ARGUMENTS

-weight *weight_value*

Specifies a cost function weight for this group. The *weight_value* must be a number between 0.0 and 100.0. The default is 1.0. A weight of 0.0 eliminates the paths in this group from cost function calculations. Do not use very small values (for example,

0.0001); smaller values can prevent **compile** from implementing small improvements to the design. If you specify **-weight** when you add members to an existing group, the new weight for the group is used.

-critical_range *range_value*

Specifies a margin of delay for *group_name* during optimization. The *range_value* must be positive or 0.0. If you do not specify a *range_value* for a group, the default is the value set with the **set_critical_range** command (the default setting is 0.0). A *range_value* of 0.0 means that only the most critical paths (the ones with the worst violation) are optimized. If you specify a nonzero *range_value*, other near-critical violating paths (one per endpoint) within that amount of the worst path are also optimized if possible. If more than one critical path to an endpoint must be optimized, use a separate **group_path** command for each distinct critical path to that endpoint. To force **compile** to optimize all violating paths, use a very large *range_value* (larger than any expected path violations).

-default

Specifies that endpoints or paths be moved to the default group and removed from the current group. You must specify **-default** unless you use **-name**; **-name** and **-default** are mutually exclusive.

-name *group_name*

Specifies a name for the group. If a group with this name already exists, the paths or endpoints are added to that group. Otherwise, a new group is created and named *group_name*. You must specify **-name** unless you use **-default**; **-name** and **-default** are mutually exclusive.

-from *from_list*

Specifies a list of names of clocks, ports, or pins that specify path startpoints. If a clock is given, all path startpoints related to that clock are implicitly included. This includes flip-flops in the transitive fanout of the clocks source pin or port, and ports that have input delay related to the clock.

-rise_from *rise_from_list*

Specifies a list the same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from *fall_from_list*

Specifies a list the same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through *through_list*

Specifies a list of path throughpoints (port, pin, or leaf cell names) of the current design. The path grouping applies only to paths that pass through one of the points in the *through_list*. If more than one object is included, the objects must be enclosed either in quotation marks (") or in braces ({}). If the **-through** option is specified multiple times, the group path setting applies to paths that pass through a member of each *through_list* in the order the lists were given. In other words, the path must first pass through a member of the first *through_list*, then through a member of the second list, and so on for every specified *through_list*. If the **-through** option is used in combination with the **-from** or **-to** options, the group path applies only if the **-from** or **-to** conditions are satisfied and the **-through** conditions are satisfied.

-rise_through *rise_through_list*

Specifies a list the same as the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than one time in a single command invocation, as with the **-through** option.

-fall_through *fall_through_list*

Specifies a list the same as the **-through** option, but applies only to paths with a falling transition at the specified objects. You can

specify **-fall_through** more than one time in a single command invocation as with the **-through** option.

-to *to_list*

Specifies a list of names of clocks, ports, or pins that specify path endpoints. If a clock is given, all path endpoints related to that clock are implicitly included. This includes flip-flops in the transitive fanout of the clocks source pin or port, and ports that have output delay related to the clock.

-rise_to *rise_to_list*

Specifies a list the same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-fall_to *fall_to_list*

Specifies a list the same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-comment *comment_string*

Allows the command to accept a comment string. The tool honors the annotation and preserves it with the SDC object so that the exact string is written out when the constraint is written out when you use the **write_sdc** or **write_script** command. The comment remains intact through the synthesis, place-and-route, and timing-analysis flows.

-priority *priority_level*

Assigns a priority to the command to resolve conflicting path selections between different **group_path** commands. By default, different **group_path** commands follow ordinary rules of priority to resolve conflicting path selections. For example, a command using **-from** and **-to** to select paths has priority over a conflicting command that selects a subset of those paths using **-through**, so the latter command is ignored for the paths that could be in either. By setting a higher priority in the command using the **-through** option, its path selection is honored, overriding the usual order of priority on paths in the intersection of the two groups. If unspecified, the priority is 0. You can set the priority for a **group_path** command to any integer 0 or greater.

DESCRIPTION

Groups a set of paths or endpoints for cost function calculations. The delay cost function is the sum of all groups (weight * violation), where violation is the amount for which setup was violated for all paths within the group. If there is no violation within a group, its cost is zero. Groups enable you to specify a set of paths to optimize even though there might be larger violations in another group. When endpoints are specified, all paths leading to those endpoints are grouped.

If a clock is specified in the *from_list* or *to_list*, all endpoints related to that clock are included in the group. The **create_clock** command automatically creates a group for a new clock with a weight of 1.0 and named the same as the clock name.

The weight of the default group is 1.0. If *weight_value* is not specified for a new group, the default is 1.0.

To undo **group_path**, use **reset_design** or **group_path -default**.

To list the path groups that are defined, use **report_path_group**. To show the maximum delay cost for each path group, use **report_constraint**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example groups all endpoints clocked by CLK1A or CLK1B into a new group called group1 that has a weight = 2.0:

```
prompt> group_path -name "group1" -weight 2.0 -to {CLK1A CLK1B}
```

The following example adds OUT1 and ff34/D to the existing path group named ADDR:

```
prompt> group_path -name "ADDR" -to {OUT1 ff34/D}
```

The following example removes OUT1 and CLK2 from existing groups and places them in the default group:

```
prompt> group_path -default -to {OUT1 CLK2}
```

This command adds all paths that first pass through either u1/Z or u2/Z and then pass through u5/Z or u6/Z into the group named blue:

```
prompt> group_path -name blue \  
-through {u1/Z u2/Z} -through {u5/Z u6/Z}
```

The following example groups all paths from inputs I1 and I2 to outputs O5 and O7 into a group named serious that has a weight of 10.0:

```
prompt> group_path -name serious -weight 10.0 \  
-from {I1 I2} -to {O5 O7}
```

The following example groups all endpoints clocked by CLK into a group with a critical range of 0.5:

```
prompt> group_path -name CLK -critical_range 0.5 -to CLK
```

The following example uses a large critical range value to optimize all paths. For example, assume that the worst violation is expected to be 100 units. Setting a critical range value larger than the worst violation causes optimization to speed up all paths if the transformations do not increase the worst violation (that is, make the worse path slower) for that group. In this example, the critical range value is set higher than the worst expected violation to ensure that all violating paths are considered during optimization:

```
prompt> group_path -name CLK -critical_range 1000.0 -to CLK
```

SEE ALSO

- create_clock(2)
- current_design(2)
- remove_path_group(2)
- report_constraint(2)
- report_path_group(2)
- reset_design(2)
- set_critical_range(2)
- set_input_delay(2)
- set_output_delay(2)

group_variable

Adds a variable to the specified variable group. This command is typically used only by the system administrator.

SYNTAX

```
status group_variable  
      group_name  
      variable_name
```

Data Types

```
group_name      string  
variable_name  string
```

ARGUMENTS

group_name

Specifies the name of the group to which the specified variable is added. The following are the existing variable *group_names*:

```
compile_variables  
edif_variables  
hdl_variables  
insert_dft_variables  
io_variables  
multibit_variables  
plot_variables  
schematic_variables  
synlib_variables  
system_variables  
suffix_variables  
view_variables  
vhdlio_variables  
write_test_variables
```

variable_name

Specifies the name of the variable to add.

DESCRIPTION

This command adds a variable to a variable group or creates a new variable group. For descriptions of all of the current variables and variable groups, see the individual variable group man pages, which you can access online from dc_shell by entering the following command:

```
prompt> help group_name
```

To list all of the currently-defined variables, use the **print_variable_group all** command. To list all of the variables in a group, use the **print_variable_group** *group_name* command.

EXAMPLES

The following example adds the variable **current_design** to the system group:

```
prompt> group_variable system current_design
```

gui_add_annotation

Adds an annotation to the layout window.

SYNTAX

```
status gui_add_annotation
[-window window_name]
[-group group_type]
[-type shape_type]
[-symbol_type symbol_type]
[-symbol_size symbol_size]
[-text text]
[-color color]
[-pattern pattern]
[-width line_width]
[-line_style line_style]
[-info_tip info_tip]
[-query_text query_text]
[-query_command query_command]
points
```

Data Types

```
window_name   string
group_type    string
shape_type    string
symbol_type   string
symbol_size   string
text          string
color         string
pattern       string
line_width    string
line_style    string
info_tip     string
query_text    string
query_command string
points       list
```

ARGUMENTS

-window *window_name*

Specifies the instance name of the layout window. If window name is omitted, the annotation will apply to all layout windows.

-group *group_type*

Specifies the annotation group. If group name is omitted, the **global** name is used.

-type *shape_type*

Specifies the type of annotation. The supported values for this option are

rect
line
arrow
polygon
polyline
text
symbol

The default is rect.

-symbol_type *symbol_type*

Specifies the type of symbol annotation. This option is only valid when annotation type is symbol. The supported values for this option are

square
x
triangle
diamond

The default is square.

-symbol_size *symbol_size*

Specifies the size of symbol annotation. An even symbol size will grow one automatically to find the appropriate center point. The default symbol size is 3.

-text *text*

Specifies the annotation text.

-color *color*

Specifies the annotation color. You can use a predefined color string, such as white, red, green, blue, yellow, and so forth, or a hexadecimal RGB value, such as #AA6633. The default annotation color is white.

-pattern *pattern*

Specifies the annotation fill pattern. The default fill pattern is none. An empty value forces the tool to use the window default value, which is usually set to solid fill.

-width *line_width*

Specifies the annotation line width. The default line width is 1.

-line_style *line_style*

Specifies the annotation line style. The default line style is none. An empty value forces the tool to use the window default value, which is usually set to solid line.

-info_tip *info_tip*

Specifies an info tip for this annotation. When the user starts the query tool in the layout, and puts the mouse cursor over this annotation the specified text will be displayed as the infoTip.

If the text starts with a '=' character the text after it will be considered to be a tcl command which, when executed, will return the query text to be displayed. The tcl command can contain special strings, which are replaced with details of the layout window and annotation, so this information can be used in the tcl procedure.

%window the layout window name

%view the layout view name

%object a collection containing the annotation

-query_text *query_text*

This option is only valid if an info tip was specified. Use this to specify a more detailed string that is displayed in the query palette when the object is selected via the query tool. If query text is not specified then the query tool will just use the info tip string.

If the text starts with a '=' character the text after it will be considered to be a tcl command which, when executed, will return the query text to be displayed. The tcl command can contain special strings, which are replaced with details of the layout window and annotation, so this information can be used in the tcl procedure.

%window the layout window name

%view the layout view name

%object a collection containing the annotation

-query_command *query_command*

This option is only valid if an info tip was specified. Use this to specify a tcl command to be run when the user clicks on the annotation.

The tcl command can contain special strings, which are replaced with details of the layout window and annotation, so this information can be used in the tcl procedure.

%window the layout window name

%view the layout view name

%object a collection containing the annotation

%x the x position clicked in design coordinates

%y the y position clicked in design coordinates

points

Specifies the points for the specified annotation type. The points are specified by a tool command language (Tcl) list in which each list element is one of the following:

- the {x y} location of a point
- a collection containing a single object that has a visual representation in the layout. In other words, it is not a net, for example. The location of the point is the origin of the object.

If a collection is specified the list element can optionally include a position type, one of:

- center : the annotation is placed at the center of the largest rectangle of the object. The largest rectangle is the rectangle with the most area which is completely contained inside the object shape.
- bbox_center : the annotation is placed at the center of the bounding box of the object.
- bbox_ll : the annotation is placed at the lower left of the bounding box of the object.
- bbox_lr : the annotation is placed at the lower right of the bounding box of the object.
- bbox_ul : the annotation is placed at the upper left of the bounding box of the object.
- bbox_ur : the annotation is placed at the upper right of the bounding box of the object.

If a bounding box position type is specified, i.e. one of 'bbox_center', 'bbox_ll', 'bbox_lr', 'bbox_ul' or 'bbox_ur', then the list element can also optionally include an x and y fractional offset from this bounding box point. The x and y offset is a fraction of the width and height of the object's bounding box respectively. For example, for a 'bbox_center' position type, an x offset of -0.5 is the left edge and an x offset of 0.5 is the right edge. Similarly a y offset of -0.5 is the bottom edge and a y offset of 0.5 is the top edge.

For annotations referring to a single-object collection, if the object is moved, the tool updates the point automatically as needed.

DESCRIPTION

This command defines a new annotation and returns a collection containing that annotation if it is successful.

EXAMPLES

Create a green rectangle annotation.

```
prompt> gui_add_annotation -window Layout.1 \
-type rect -color green {{200 200} {777 777}}
```

Create a red polyline annotation in the group Test1.

```
prompt> gui_add_annotation -window Layout.2 \
-group Test1 -type polyline -color red \
-width 2 {{123 456} {789 12} {200 200}}
```

Create a line between two cells A and B.

```
prompt> gui_add_annotation -window Layout.1 \
-type line [list [get_cells A] [get_cells B]]
```

Create a line between the center of largest rectangle of cell A and the center of the largest rectangle of cell B.

```
prompt> gui_add_annotation -window Layout.1 \
-type line [list [list [get_cells A] center] [list [get_cells B] center]]
```

Create a line between the bottom left of cell A's bounding box and the middle of the top of cell B's bounding box.

```
prompt> gui_add_annotation -window Layout.1 -type line \
[list [list [get_cells A] bbox_ll] [list [get_cells B] bbox_center 0.0 0.5]]
```

Create a blue rectangle which displays the annotation points using the `annotation_query` tcl procedure.

```
prompt> proc annotation_query { object window view } { \
return [get_attribute $object points] \
}
```



```
prompt> gui_add_annotation -window Layout.1 -type rect -color blue \  
-query_text {=annotation_query %object %window %view} -info_tip {annotation points} \  
{{110 110} {160 160}}
```

SEE ALSO

gui_remove_annotations(2)
gui_remove_all_annotations(2)

gui_bin

Organizes a collection of objects into bins.

SYNTAX

```
string gui_bin
  -clct Clct
  [-attr Attribute | -cmd Command]
  [-lower_bound Lower_Bound]
  [-lower_bound_strict]
  [-upper_bound Upper_Bound]
  [-upper_bound_strict]
  [-boundary Boundary]
  [-num_bins Number_of_Bins
   | -bin_range Range_per_Bin]
  [-underflow]
  [-overflow]
  [-nice_level Nice_Level]
  [-small_is_good]
  [-exact_binning]
  [-ignore_values Ignore_List]
  [-bar_brush Brush_Pattern]
  [-create_slct_buses]
  [-filter_cmd Filter_Command]
  [-numBin numBins]
  [-return_values]
  [-slct_targets Selection_Targets]
  [-slct_targets_operation Selection_Operation]
  [-value_list Element_Value_List]
```

Data Types

<i>Clct</i>	collection
<i>Attribute</i>	string
<i>Command</i>	string
<i>Lower_Bound</i>	float
<i>Upper_Bound</i>	float
<i>Boundary</i>	float
<i>Number_of_Bins</i>	integer
<i>Range_per_Bin</i>	float
<i>Nice_Level</i>	integer
<i>Ignore_List</i>	list
<i>Brush_Pattern</i>	string
<i>Filter_Command</i>	string
<i>numBins</i>	integer
<i>Selection_Operation</i>	string
<i>Element_Value_List</i>	list

ARGUMENTS

-clct *Clct*

Specifies the collection of objects that the tool partitions into bins.

-attr *Attribute*

Specifies the attribute that the tool uses to get a floating point value for each object in the collection. The tool places the objects in bins based on these attribute values.

The **-attr** and **-cmd** options are mutually exclusive.

-cmd *Command*

Specifies the tool command language (Tcl) command that the tool uses to get a floating point value for each object in the collection. The tool places the objects in bins based on these values.

The tool replaces all occurrences of %clct in the command with a collection that contains, as its only element, the object used to compute the floating point value for the object. If the tool cannot store the object in a collection, it uses the string "Error" instead.

The tool replaces all occurrences of %string in the command with the name of the object. If the tool cannot compute the name for an object, it uses the string "Error" instead.

The **-cmd** and **-attr** options are mutually exclusive.

-lower_bound *Lower_Bound*

Defines the lower bound for the range of floating point values that the tool uses to place the objects in bins.

If you specify the **-underflow** option, the tool places all the objects with a value that is less than *Lower_Bound* in a separate underflow bin. If you do not specify the **-underflow** option, the tool discards these objects.

-lower_bound_strict

Forces the tool to use the **-lower_bound** option value as the lower bound of the bin with the smallest values, not including the underflow bin.

If you do not specify the **-lower_bound_strict** option, the tool sets the lower bound of the bin with the smallest values, not including the underflow bin, to the smallest value that is greater than or equal to the value specified by the **-lower_bound** option.

-upper_bound *Upper_Bound*

Defines the upper bound for the range of floating point values that the tool uses to place the objects in the bins.

If you specify the **-overflow** option, the tool places all the objects with a value that is greater than *upper_Bound* in a separate overflow bin. If you do not specify the **-overflow** option, the tool discards these objects.

-upper_bound_strict

Forces the tool to use the **-upper_bound** option value as the upper bound of the bin with the largest values, not including the overflow bin.

If you do not specify the **-upper_bound_strict** option, the tool sets the upper bound of the bin with the largest values, not including the overflow bin, to the largest value that is less than or equal to the value specified by the **-upper_bound** option.

-boundary *Boundary*

Specifies the boundary value for one of the bins, not including the underflow and overflow bins. This value must fall within the range of values covered by the bins.

If the relevant value distribution has a specific value where the quality of the values change, such as 0 for slack, you can use this option to make sure that no bin contains values on both sides of the boundary value.

-num_bins *Number_of_Bins*

Specifies the number of bins that this command generates, not counting underflow and overflow bins. The default value is 8.

The **-num_bins** and **-bin_range** options are mutually exclusive. If you specify both of these options, the tool ignores the **-bin_range** option. If you do not specify either of these options, the default value is 8.

-bin_range *Range_per_Bin*

Specifies the size of the value range represented by each bin.

The **-bin_range** and **-num_bins** options are mutually exclusive. If you specify both of these options, the tool ignores the **-bin_range** option. If you do not specify either of these options, the default value is 8.

-underflow

Collects all the objects with values less than the lower bound value specified by the **-lower_bound** option, and places them in a separate underflow bin. Otherwise, these objects are discarded.

-overflow

Collects all the objects with values greater than the upper bound value specified by the **-upper_bound** option, and places them in a separate overflow bin. Otherwise, these objects are discarded.

-nice_level *Nice_Level*

Determines how round the boundaries of the bins are. The default value is 10. For rounder boundaries, specify a higher value. If *Nice_Level* is -1, the tool does not attempt to make the boundaries round.

-small_is_good

Sorts the values in every bin in order of decreasing values. By default, the tool sorts the objects in order of increasing values.

In addition, bins with values that are less than the value specified by the **-boundary** option are colored green, and bins with values that are greater than this value are colored red. This is opposite from the default color markings.

-exact_binning

Sets exact boundaries on the left border of the furthest left bin and the right border of the furthest right bin. By default, the tool can adjust these limits to create bins with equal widths.

-ignore_values *Ignore_List*

Discards the objects with values specified in *ignore_List*.

-bar_brush *Brush_Pattern*

Specifies the brush pattern for painting bars in a histogram based on the result of the command. The *brush_Pattern* can be one of the following values:

NoBrush	SolidPattern	Dense1Pattern
Dense2Pattern	Dense3Pattern	Dense4Pattern
Dense5Pattern	Dense6Pattern	Dense7Pattern
HorPattern	VerPattern	CrossPattern
BDiagPattern	FDiagPattern	DiagCrossPattern

-create_slct_buses

Creates selection buses to return the objects instead of collections.

-filter_cmd *Filter_Command*

Specifies a Tcl command that the **gui_bin** command uses to determine whether a collection in a bin should be filtered out.

The *filter_Command* format is

```
cmd %clct %attr %value
```

where *cmd* is a Tcl command or procedure, *%clct* is a collection, *%attr* is the name of the attribute used to create the bins, and *%value* is the maximum value for *%clct*. If the command returns a nonzero value, **%clct** is not placed in a bin.

-numBin *numBins*

This option is obsolete. Do not use it. Use the **-num_bins** option instead.

-return_values

Causes the **gui_bin** command to return the values of elements.

-slct_targets *Selection_Targets*

Specifies a list of selection containers to store the objects.

-slct_targets_operation *Selection_Operation*

Specifies an operation applied to the selection containers.

-value_list *Element_Value_List*

Specifies values for the elements in a list. The **gui_bin** command uses these values to create the bins.

DESCRIPTION

The **gui_bin** command creates a list of bins from a collection of objects that you specify with the **-clct** option. The command determines a floating point value for each of the objects by using either an attribute that you specify with the **-attr** option or a Tcl command that you specify with the **-cmd** option.

Each bin is represented by a list that contains the left boundary, the right boundary, the number of objects in the bin, a collection containing the objects in the bin, and the value green or red.

The widths of the bins are equal by default, which means that in some cases the furthest left and furthest right bins might extend the specified boundaries in order to get nice values for the boundaries. If you specify the **-exact_binning** option, the left border of the furthest left bin and the right border of the furthest right bin are set exactly to the specified boundaries.

The collections are sorted with ascending floating point values by default, and a bin is marked green only if the objects it contains have values greater than or equal to the value specified by the **-boundary** option. If you specify the **-small_is_good** option, the values in the collections are sorted with descending floating point values, and a bin is marked red only if the objects it contains have values greater than or equal to the value specified by the **-boundary** option.

The bins cover the complete range of floating point values by default. You can define the covered range by using the **-lower_bound**, **-lower_bound_strict**, **-upper_bound**, and **-upper_bound_strict** options. In this case, you can collect objects with floating point values that are too small or too big and place them into separate underflow and overflow bins by specifying the **-underflow** and **-overflow** options.

You can specify a list of objects with floating point values that can be ignored by using the **-ignore_values** option.

You can specify a bin range size that results in a very large number of bins by using the **-bin_range** option. However, this results in a histogram that is difficult to read and that might take a long time to compute and draw.

The preference variable **hist_max_num_bins** in the Histogram category allows you to set the maximum number of histogram bins. The default value is 100. If a specified range size value causes the number of bins to exceed the value set for this variable, the tool issues an error message and does not create a histogram. The error message includes information about the data range to aid you in specifying a better **-bin_range** value. This variable is used only in conjunction with the **-bin_range** option and is not used with the **-num_bins** option.

EXAMPLES

The following example creates a list of bins containing the 100 worst paths according to their slack distribution:

```
prompt> set binInfo [gui_bin -clct [get_timing_paths \  
-nworst 100 -max_paths 100] -cmd "get_attribute %clct slack"]  
prompt> set bins [lindex $binInfo 0]
```

SEE ALSO

gui_change_highlight

Manipulate the set of globally highlighted objects.

SYNTAX

```
string gui_change_highlight  
  [-add | -remove | -toggle]  
  [-color color_id | -all_colors]  
  [-collection clct]
```

Data Types

```
color_id  string  
clct      collection
```

ARGUMENTS

-add

Use -add to highlight a collection of objects in a color specified with color. If -color is not specified, the current color is used. You cannot use -all_colors with -add. You must specify a collection with -collection. If -remove or -toggle are not specified, then -add is implied.

-remove

Use -remove to remove highlighting from objects. Use -collection to remove highlighting from specific objects. Use -color to remove highlighting from objects of a specific color. Use -all_colors to remove all highlighting. If you specify a collection, you cannot specify colors.

-toggle

Use -toggle to toggle the highlight state of a collection of objects. You must use -collection with -toggle, and you cannot use -color or -all_colors. Toggling an object that is highlighted will cause it to no longer be highlighted. Toggling an object that is not highlighted will cause it to be highlighted with the current color. If no operation is specified, then -add is assumed.

-color *color_id*

Specifies the color to be used for the highlight operation. This is mutually exclusive with -all_colors. The allowed colors are blue, light_blue, yellow, purple, light_purple, orange, light_orange, red, light_red, green, and light_green. If neither -color nor -all_colors is specified, then the current highlight color is assumed. You cannot use -color with -toggle.

-all_colors

This option is only valid with -remove. Use this option to clear all highlight colors.

DESCRIPTION

The `gui_change_highlight` command is used to operate on the set of highlighted objects. Objects can be added or removed from the set, or their presence can be toggled as described above. The set has a current color that can be modified with `gui_set_highlight_options(2)`. The current color is used as needed when no color is specified.

The tool provides eleven built in highlight colors with the names yellow, orange, red, green, blue, purple, light_orange, light_red, light_green, light_blue, and light_purple.

EXAMPLES

Highlight in green all cells with names matching `foo*`.

```
prompt> gui_change_highlight -color green -collection [get_cells foo* -hier -all]
```

Remove highlights from all objects in the collection returned by the `get_cells` command.

```
prompt> gui_change_highlight -remove -collection [get_cells foo* -hier -all]
```

Clear all objects with blue highlights.

```
prompt> gui_change_highlight -remove -color blue
```

Clear objects with the current highlight color.

```
prompt> gui_change_highlight -remove
```

Clear all highlights.

```
prompt> gui_change_highlight -remove -all_colors
```

Toggle the highlight state of all objects in a collection returned by the `get_cells` command.

```
prompt> gui_change_highlight -toggle -collection [get_cells foo*]
```

SEE ALSO

`gui_get_highlight(2)`
`gui_get_highlight_options(2)`
`gui_set_highlight_options(2)`

gui_close_window

Close the specified window

SYNTAX

```
status gui_close_window
{ -window window_id | -type window_type | -all }
[ -exact_type_match ]
```

Data Types

```
window_id    string
window_type  string
```

ARGUMENTS

-window *window_id*

Specifies the window name id.

Any matching window of this name will be closed.

This option precludes the *-type* and *-all* options.

-type *window_type*

Specifies the window type id. Any matching window of this type will be closed.

If the *-exact_type_match* options is not specified then types derived from this window type will also be closed. Derived types are created with the **gui_create_window_type** command.

This option precludes the *-window* and *-all* options.

-all

Specifies that we want to close all the windows.

This option precludes the *-window* and *-type* options.

-exact_type_match

Specifies that types derived from the specified type are excluded when the *-type* option is specified.

Note: this option should only be used with the *-type* option.

DESCRIPTION

This command closes the specified window(s).

Windows can be specified by name, type or as all windows.

EXAMPLES

The following examples will create a toplevel window and a child layout view window, then close the command layout window.

```
shell> set top [gui_create_window -type TopLevel]
shell> set x [gui_create_window -type Layout -parent $top]
shell> gui_close_window -window $x
```

SEE ALSO

- gui_create_window(2)
- gui_exist_window(2)
- gui_show_window(2)
- gui_get_window_types(2)
- gui_get_window_ids(2)
- gui_set_active_window(2)
- gui_get_current_window(2)

gui_create_attrgroup

Creates a group of attributes for an object type.

SYNTAX

```
status gui_create_attrgroup  
-class design_object  
-name name  
[-attr_list {{attr_1}...{attr_n}}]
```

Data Types

<i>design_object</i>	string
<i>name</i>	string
<i>attr_1</i>	string
<i>attr_n</i>	string

ARGUMENTS

-class *design_object*

Specifies the type of design object.

-name *name*

Specifies the name of the attribute group to be created for the specified object type.

-attr_list {{*attr_1*}...{*attr_n*}}

Lists and orders the attributes to be included in the group.

DESCRIPTION

The **gui_create_attrgroup** command creates a new attribute group for a specific type of design object. The command adds the attribute group to the display in GUI tools, such as the Properties dialog box, List views, layout view InfoTips, and other graphic windows.

An attribute can be assigned to more than one attribute group at the same time.

The order of the attributes in an attribute group is important and is set by the **-attr_list** option when you create the group. Use the **gui_update_attrgroup** command if you need to change or reorder the attribute list for an attribute group.

EXAMPLES

```
prompt> gui_create_attrgroup -class Cell -name "Hierarchy" \  
-attr_list { "Name" "Hierarchical" "Owning Cell"  
"isPhysConstraint" "Hard Macro" "Num Children" "Num Nets"  
"Num Pins" "Utilization" "Area" "Aspect Ratio" "Min Aspect"  
"Max Aspect" "Min Area" "Max Area" "Area / 1M" "Hier Req Area / 1M"  
"Reference" "Logical Name" "FullName" }
```

```
prompt> gui_create_attrgroup -class Cell -name "Edit" \  
-attr_list { "Name" "isPhysConstraint" "Orientation"  
"Location" "Bounding Box" "Fixed Location" "Is Resizable"  
"Hard Macro" "Placed" "Is IO" "Off Litho Grid" "FullName" }
```

```
prompt> gui_create_attrgroup -class Pin -name "Timing" \  
-attr_list { "Name" "Is Clock" "Slew" "slack_max" }
```

```
prompt> gui_create_attrgroup -class Net -name "Timing" \  
-attr_list { "Name" "Lumped C" "Total Lumped C"  
"Lumped Resistance" }
```

SEE ALSO

```
gui_delete_attrgroup(2)  
gui_list_attrgroups(2)  
gui_update_attrgroup(2)
```

gui_create_category_rule

Creates a category rule for automatic generation of one or more object categories.

SYNTAX

```
string gui_create_category_rule  
  [-name rule_name]  
  [-filter filter_spec]  
  -category category_spec  
  [-builtin]
```

Data Types

```
rule_name      string  
filter_spec    string  
category_spec  string
```

ARGUMENTS

-name *rule_name*

Specifies the name of the category rule to be created. If this option is omitted, a unique new rule name is automatically generated.

-filter *filter_spec*

Specifies the filter. The basic form of a filter conditional expression is a series of relations joined together with && (and), || (or), and ! (not) operators. Parentheses are used to override precedence. The basic relation is either a Boolean attribute or a comparison of one non-Boolean attribute with another or with a constant value using relational operators.

For example, a filter expression can have any of the following forms:

- **-filter {endpoint_clock_is_propagated}** (Boolean attributes)
- **-filter {(slack < 0)}** (Comparison with a literal number)
- **-filter {(path_group.full_name == "inputs")}** (Comparison with a literal string)
- **-filter {(startpoint_clock.full_name == endpoint_clock.full_name)}** (Comparison of attributes)

Some object attributes have values that are object collections of size one. For those attributes, "dot notation" can be used to access attributes of the object in the collection. The following relational operators are supported:

- **==** (Equal)
- **!=** (Not equal)
- **==~** (Matches pattern)

- `!~` (Does not match pattern)
- `&~` (Contains all elements)
- `|~` (Contains some elements)
- `>` (Greater than)
- `<` (Less than)
- `>=` (Greater than or equal to)
- `<=` (Less than or equal to)

The matching operators are used to match wildcard regular expressions. The containment operators are used to compare space-separated set or list attributes.

-category *category_spec*

Specifies the category. The category specification can be fixed literal text, or it can include an optional object attribute name enclosed in angle brackets; for example, **-category <path_type>**. Additional literal text is optionally allowed to the left and to the right of the angle brackets; for example **-category {Path Type: <path_type>}**. Some object attributes have values that are object collections of size one. For those attributes, "dot notation" can be used to access attributes of the object in the collection; for example, **-category <startpoint_clock.full_name>**.

When a category rule is "executed" later in the GUI, a separate category (bin of objects) is generated for each unique value of the attribute specified by the **-category** option. (The GUI "category-rule-execution engine" examines attribute values for an input set of objects.)

-builtin

Indicates that the rule being created belongs to the built-in group of category rules. If this option is omitted, the rule being created does not belong to the built-in group.

The built-in group of category rules is a group of category rules that are created before any non-built-in rules. This group of rules includes both built-in rules created by the tool (at tool startup time) and built-in rules that you create before creating any non-built-in rules.

DESCRIPTION

This command creates a category rule. If rule creation is successful, the command returns the name of the newly-created category rule; otherwise, it returns an empty string. After a category rule has been created, it can be used later (at "category-rule execution time" in the GUI) with other rules to generate one or more object categories.

EXAMPLES

This example creates a simple category rule:

```
prompt> gui_create_category_rule -name pathgroups \
      -category <path_group.full_name>
pathgroups
```

This example creates a simple filtering rule:

```
prompt> gui_create_category_rule -name {Negative Slack} \  
       -category {Failing Paths} -filter {slack < 0}  
Negative Slack
```

This example creates a regular expression filtering rule:

```
prompt> gui_create_category_rule -name {ATPG Sessions} \  
       -category {ATPG Sessions} -filter {session_name =~ "atpg*"}  
Negative Slack
```

This example creates a set or list containment filtering rule:

```
prompt> gui_create_category_rule -name {Through CPU and CTRL} \  
       -category {Through CPU and CTRL} -filter { blocks &~ "CPU CTRL"}  
Negative Slack
```

SEE ALSO

`gui_list_category_rules(2)`
`gui_remove_category_rules(2)`

gui_create_menu

Create a menu item for the toplevel menubar or a context menu item. A menu hierarchy, based on the menu name, may be created as necessary to host the menu item.

SYNTAX

```
string gui_create_menu -menu HierMenuName
<-tcl_cmd TclCmd[-separator]-heading headingText>
[-enable_cmd EnableCmd]
[-get_state_cmd GetStateCmd]
[-icon IconFilePath]
[-hot_key HotKey]
[-tooltip ToolTip]
[-help_string HelpStr]
[-anchor_item AnchorItem]
[-anchor_offset AnchorOffset]
[-position MenuPos]
[-window_type WindowTypeName]
[-root ContextMenuRoot]
[-reference RefMenuItem]
```

```
HierMenuName String
TclCmd String
EnableCmd String
GetStateCmd String
IconFilePath String
HotKey String
ToolTip String
HelpStr String
AnchorItem String
AnchorOffset Integer
MenuPos Integer
WindowTypeName String
ContextMenuRoot String
RefMenuItem String
```

ARGUMENTS

-menu *HierMenuName*

HierMenuName specifies the hierarchical menu name of the menu item to create. If a level of the hierarchy does not exist in the menu structure it will be created automatically by this command. The "name" of a menu is a string which specifies the hierarchy of menu labels connected with '->'. For example, File->Quit specifies the Quit menu item under the File menu. These names are allowed to also specify the mnemonic for the menu text which is specified with an embedded '&', but they are not required to do

so.

-tcl_cmd *TclCmd*

TclCmd specifies the tcl command to execute when the menu item is selected. This option is mutually exclusive with the -separator and -heading options.

-separator

This option creates a menu item separator. It is mutually exclusive with the -tcl_cmd and -heading options.

-heading *headingText*

This option creates a text heading in the menu with the specified **headingText**. It is visually distinct from actual menu items, and it is not selectable like an actual menu item is. This is used to provide general grouping information without requiring an additional level of sub menu. This option is mutually exclusive with the -separator and -tcl_cmd options.

-enable_cmd *EnableCmd*

EnableCmd specifies the tcl command to evaluate whether the menu item should be enabled or disabled. The tcl command should return "true" or "1" to enable, and return "false" or "0" to disable. If the enable_cmd is not specified then the menu item will always be enabled.

-get_state_cmd *GetStateCmd*

GetStateCmd specifies the tcl command that will be executed to determine whether the menu item should be in an on (checked or pushed-in) or off (unchecked or normal) state. Most menu items don't have persistent state, and for those the get_state_cmd should not be specified. This tcl command returns "true" or "1" for the "on" state, and returns "false" or "0" for the "off" state.

-icon *conFilePath*

IconFilePath specifies the absolute pathname of the icon file. The file should be in .xpm format. Typical application-supplied icons are 16x16 pixels in size and use a transparent background (color 'None' in xpm format).

-hot_key *HotKey*

HotKey specifies the hotkey (accelerator) for the menu item.

-tooltip *ToolTip*

ToolTip specifies the tooltip for the menu item.

-help_string *HelpStr*

HelpStr specifies the help string for the menu item. The help string is displayed in the status bar.

-anchor_item *AnchorItem*

AnchorItem specifies the existing menu item that will be used as the anchor position for this new menu item. This option is used together with the -anchor_offset option to determine the location of the new menu item. The -position option cannot be used along with the -anchor_item and -anchor_offset options.

-anchor_offset *AnchorOffset*

AnchorOffset specifies the offset from the anchor item position that will be used to determine the location of the new menu item in the menu hierarchy. The offset must be a positive or negative integer, and not zero. If positive, the new menu item will be placed below the anchor item by the specified offset. If negative, it will be placed above the anchor item position by the absolute value of the specified offset. The -position option cannot be used along with the -anchor_item and -anchor_offset options.

-position *MenuPos*

MenuPos specifies the absolute menu position for the menu item within the parent popup menu. The -position option cannot be

used with the `-anchor_item` and `-anchor_offset` options.

-window_type *WindowTypeName*

WindowTypeName specifies the type of top level window whose menu bar is to be modified. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable `gui_default_window_type(3)` specifies the default window type for the application.

-root *ContextMenuRoot*

ContextMenuRoot specifies the context menu root to which the menu item being defined will be added. A context menu root groups menu items that share the same context menu root. The context menu root is used for context menu. Context menu, is usually brought up by right clicking on a window and is used to provide context-sensitive features. (Note the difference between a context menu root and the menu root use by a toplevel window's menu bar .)

-reference *RefMenuItem*

RefMenuItem specifies the menu item under the toplevel window's menu bar, which will be used to define the behavior of the menu item being defined. Often used with the `-root` option to help associate a context menu item with an already defined menu entry under the menu bar. Used with the `-window_type` option to completely specify the reference menu item under the menu bar. If the `-window_type` option is not specified, the default toplevel window type as specified with the tcl variable `"::gui_default_window_type"` will be used.

DESCRIPTION

This command creates a menu item for the menu bar of a top level window. A menu hierarchy, based on the menu name, may be created as necessary to host the menu item. The menu hierarchy separator is denoted by `"->"`. These settings are stored either in a user-specific setup file or one that is shared across the installation. The gui initialization sequence will load builtin system configuration, and then apply the installation specific customizations, and finally the user's customizations. The installation specific setup file is specified by the variable `gui_custom_setup_file` and will have a default empty value. This can be overridden in the `.synopsys_dc.setup` file if it is desired. If the file specified by this variable exists then it will be loaded after the gui is initialized. The user setup file is `~/synopsys_icc_gui.tcl` for IC Compiler.

EXAMPLES

The following simple example adds a new toplevel popup menu *&Favorite* in the menubar and add a submenu named *Preferences* then add an item named *My &Item* to that submenu. This menu item will be enabled if *enable_my_item* is set to 1 and disabled otherwise. A hotkey and tooltip are also provided.

```
set enable_my_item 1
proc enable_my_item {} {
    if { $::enable_my_item == 1 } {
        return 1
    }
    return 0
}
gui_create_menu -menu "&Favorite->Preferences->My &Item" \
    -tcl_cmd "echo {executing My Item}" \
    -enable_cmd "enable_my_item" \
    -hot_key "Ctrl+N" \
    -tooltip "short tooltip description for My Item" \
```

```
-help_string "long status bar description for My Item" \  
-icon_file "/syn/sparc/my_item.xpm"
```

The following IC Compiler example illustrates creating a new menu item in the context menu of the layout window which will refer to the Zoom Fit Selection menu item from the Layout window main menu bar. This assumes that the tcl variable "gui_default_window_type" is set to the ICC window type "LayoutWindow" and thus it is not necessary to specify the -window_type option with the value of "LayoutWindow".

```
gui_create_menu -menu "Zoom Fit Selection" \-root layout_view_menu_root \-reference "View->Zoom->Zoom Fit Selection"
```

The following example create a new Tcl-based menu entry in the context menu which invokes a procedure # called do_my_thing.

```
gui_create_menu -menu "Do My Thing" \-root layout_view_menu_root \-tcl "do_my_thing"
```

The following IC Compiler example illustrates adding a context menu item to the path schematic context menu which refers to a menu item in the Main window. This example assumes that there is a context menu root called "path_schematic_contextmenu_root", a hierarchical menu item in the toplevel menu bar of the window type "MainWindow" with the hierarchical menu name "Schematic->Delete Selected".

```
gui_create_menu -menu "Remove Selected" \-root path_schematic_contextmenu_root \-reference "Schematic->Delete  
Selected" -window_type MainWindow
```

SEE ALSO

```
gui_set_hotkey(2)  
gui_report_hotkeys(2)  
gui_delete_menu(2)  
gui_create_toolbar(2)  
gui_delete_toolbar(2)  
gui_create_toolbar_item(2)  
gui_delete_toolbar_item(2)  
gui_get_menu_roots(2)  
gui_default_window_type(3)  
gui_custom_setup_files(3)
```

gui_create_pref_category

Create a preference category.

SYNTAX

```
string gui_create_pref_category  
-category Category
```

ARGUMENTS

-category *Category*

Category specifies the category to create.

DESCRIPTION

This command creates a preference service category with the specified name, which can be used to store and categorize key-value pairs that are created with the **gui_create_pref_key** command. If successful, the command returns the name of the category that is created.

EXAMPLES

The following example create a user-defined category *some_category* for storing preferences and then create a boolean key *some_key* under that category with the initial value of false.

```
gui_create_pref_category -category some_category  
gui_create_pref_key -category my_category -key some_key -value_type bool -value false
```

SEE ALSO

```
gui_create_pref_key(2)  
gui_set_pref_value(2)  
gui_get_pref_value(2)  
gui_remove_pref_key(2)
```

```
gui_get_pref_keys(2)
gui_exist_pref_category(2)
gui_exist_pref_key(2)
gui_update_pref_file(2)
```

gui_create_pref_key

Create a preference key.

SYNTAX

```
string gui_create_pref_key  
-key Key  
-value_type ValueType  
-value Value  
[-category Category]  
[-keep_value_if_exist]  
[-read_only]  
[-description Description]  
[-min Minimum_Value]  
[-max Maximum_Value]  
[-legal_value_list Value_ListfP]  
[-string_to_int_map Value_Map]  
[-save_on_exit Boolean_Value]
```

ARGUMENTS

-key *Key*

Key specifies the name of the key to create.

-value_type *ValueType*

ValueType specifies the data type of the value of the key. It must be one of [bool | integer | double | color | string].

-value *Value*

Value specifies the value of the key. The value should be set appropriately in accordance with its value type. The **bool** type accepts [true, false, 0, 1, on, off]. The **color** type accepts a named color, eg. "red" or a string specifying the color in this format "#RRGGBB", for example, "#0000FF".

-category *Category*

Category specifies the category that the key belongs to. If not specified, a default category will be assigned.

-keep_value_if_exist

This Boolean option specifies to keep current value if the key already exists. The default is that the key will be assigned the value given in the **-value** option.

-read_only

If given, this flag specifies that the preference key value is read-only and its value cannot be set after the key has been created.

-description *Description*

If given, the given description string with the meaning of the key is assigned to the preference key.

-min *Minimum_Value*

If given, specifies the minimum value for an integer or double type preference key.

-max *Maximum_Value*

If given, specifies the maximum value for an integer or double type preference key.

-legal_value_list *Value_List**P*

If given, this option specifies discrete valid values for an integer, double or string type preference key. Input the argument using Tcl list syntax: {{prefVal} ...}

-string_to_int_map *Value_Map*

If given, provides a map from a string key value to an integer value for preferences of integer type. Input the argument as a Tcl list of Tcl list where the internal list is a pair of the string value and the integer value: {{stringval intval} ...}

-save_on_exit *Boolean_Value*

If given, specifies whether the preference key value is saved to the user preference file upon existing the application. Acceptable argument is either **true** or **false**.

DESCRIPTION

This command creates a preference key with the specified key name. This key will have the specified value type and value, and it will be created under the specified category. If the category is not specified, the key will belong to the default category. Returns the value of the preference key.

EXAMPLES

The following example create a user-defined category *some_category* for storing preferences and then create an integer key *some_key* under that category with the initial value of 200.

```
gui_create_pref_category -category some_category
gui_create_pref_key -category my_category -key some_key -value_type integer -value 200
```

SEE ALSO

```
gui_create_pref_category(2)
gui_set_pref_value(2)
gui_get_pref_value(2)
gui_remove_pref_key(2)
```

```
gui_get_pref_keys(2)
gui_exist_pref_category(2)
gui_exist_pref_key(2)
gui_update_pref_file(2)
```

gui_create_schematic

Creates a schematic or symbol view.

SYNTAX

```
string gui_create_schematic  
  [-hierarchy]  
  [-no_bus]  
  [-symbol_view]  
  [-reference]  
  [-schematic_view]
```

ARGUMENTS

-hierarchy

Generates a schematic for each design in the current design.

-no_bus

Generates the schematic or symbol view without busing. Related pins and nets are not represented by a bused object in the new schematic.

-symbol_view

Generates a symbol view of the selected designs and cells. If nothing is selected, generate a symbol view of the current design.

-reference

Generates a schematic view of the reference designs of the current instance.

-schematic_view

Generates a schematic view of the currently selected designs and cells.

DESCRIPTION

The command generates a schematic or symbol view of the currently selected designs and cells.

EXAMPLES

The following example generates a bused schematic view, a bused symbol view, an unbused schematic view, and an unbused symbol view of the hierarchical cell named RAM_1.

```
prompt> change_selection [find cell RAM_1]  
prompt> gui_create_schematic -schematic_view  
prompt> gui_create_schematic -symbol_view  
prompt> gui_create_schematic -schematic_view -no_bus  
prompt> gui_create_schematic -symbol_view -no_bus
```

SEE ALSO

`change_selection(2)`

gui_create_task

Create a task.

SYNTAX

```
gui_create_task
  -name      TaskName
  -item_root ItemRootName
  [-default]
  [-icon string]
  [-menu_string string]
  [-help_string string]
```

ARGUMENTS

-name *TaskName*

TaskName specifies the name of the new task. Task names must be unique.

-default

This option specifies that this command will be the default task set as the current task. If another task is later created with this option, the the later setting overrides any previous settings.

DESCRIPTION

This command creates a task. A task defines the user task context for the whole application. Tasks may be used to configure menus and also to present a hierarchical list of task items to the user to navigate in the Task Assistant. The hierarhical list of task items is defined with the `gui_create_task_item` command.

EXAMPLES

The following simple example creates a new task, then defines a Tk widget task page, and an HTML task page, then creates task items referencing the task pages.

```
gui_create_task -name MyAppTask -item_root MyAppTaskRoot
```

```
gui_create_tk_task_page -name InputFormPage -create_command create_widget_proc
```

```
gui_create_html_task_page -name AppDocument -path ./MyAppDocument.html
```

```
gui_create_task_item -name "Application->Input Form" -task MyAppTask -page InputFormPage
```

```
gui_create_task_item -name "Application->Documentation" -task MyAppTask -page AppDocument
```

SEE ALSO

`gui_create_task_item(2)`

`gui_create_tk_task_page(2)`

gui_create_task_item

Create an item in the task tree to access in the Task Assistant.

SYNTAX

```
gui_create_task_item
  -name      ItemName
  -task TaskName | -item_root ItemRootName
  -page      PageName
  [-search_terms TermsList]
```

ARGUMENTS

-name *ItemName*

ItemName specifies the hierarchical name given to the task item being created. This is also the text displayed to the user in the Task Assistant.

-task *TaskName*

TaskName specifies the task in which this task item is created. If a task name is used to create the task item, the item is inserted into the task item root associated with the task in the create_task command. If other tasks use the same item root, they will also see the new task item created with this command. This option is mutually exclusive with the -item_root option. One must be specified but not both.

-item_root *ItemRootName*

ItemRootName specifies the task item root in which this task item is created. Item root names are associated with task names with the gui_create_task command and allows multiple tasks to share a single task item root. This option is mutually exclusive with the -item_root option. One must be specified but not both.

-page *PageName*

PageName specifies the page content factory associated with this task item. When the user selects the task item, the page specified by this option will be constructed and shown in the Task Assistant.

-search_terms *TermsList*

TermsList specifies additional terms that can be matched by the command search feature to find this task item. For example, if the task page allows the user to specify values for input parameters to a command, you may want to associate the name of that command as additional search term for the task item.

DESCRIPTION

This command creates a task item in the given task. Task items are defined in a hierarchy with hierarchy levels delimited by the separator string "->". The hierarchy of task items are shown in the task item navigation tree in the Task Assistant.

When the user selects a leaf item in the task navigation tree, the Task Assistant will show the task page associated with the item by this command. Currently, the pages can either be constructed from an HTML document or from a Tk widget factory.

EXAMPLES

The following simple example adds a new task, then defines a Tk widget task page, and an HTML task page, then creates task items referencing the task pages.

```
gui_create_task -name MyAppTask -item_root MyAppTaskRoot
```

```
gui_create_tk_task_page -name InputFormPage -create_command create_widget_proc
```

```
gui_create_html_task_page -name AppDocument -path ./MyAppDocument.html
```

```
gui_create_task_item -name "Application->Input Form" -task MyAppTask -page InputFormPage
```

```
gui_create_task_item -name "Application->Documentation" -task MyAppTask -page AppDocument
```

SEE ALSO

`gui_create_task(2)`

`gui_create_tk_task_page(2)`

gui_create_tk_palette_type

Creates a user configurable main window palette type.

SYNTAX

```
string gui_create_tk_palette_type
    {-type type_name}
    {-title palette_title}
    {-window_types window_type_list}
    {-create_command tcl_command}
    [-dock_edge left | right | top | bottom]
    [-icon string]
```

Data Types

<i>type_name</i>	string
<i>palette_title</i>	string
<i>window_type_list</i>	string
<i>tcl_command</i>	string

ARGUMENTS

-type *type_name*

Defines a name for the new palette type. If the specified name is not unique, an error occurs.

-title *palette_title*

Defines a title string to be used to describe palettes of this type. For example, context menus which control palette visibility use the specified title.

-window_types *window_type_list*

Specifies the types of main windows that contain the new palette type. After a palette type is defined, any new main window of the types specified are created with a palette.

-create_command *tcl_command*

Specifies a command to be called each time a main window of a type specified with *window_types* is created. The command is responsible for populating the palette with Tk control widgets. The command is called with two arguments: the name of the Tk widget to be populated and the name of the new main window.

-dock_edge left | right | top | bottom

Specifies the default dock edge to use when palettes of this type are created. If palettes are repositioned using the user interface, the new positions override the default dock position the next time the application is run.

DESCRIPTION

This command defines a type of main window palette which contains a Tk widget. An instance of each defined palette type is automatically created each time a main window of any of the specified types is instantiated. During the palette's initialization, the user's Tcl command is called to provide an opportunity for defining Tk widgets within the new palette. Creating a palette type does not create a palette. It only specifies the parameters to be used to construct palettes of that type when main windows are subsequently created.

SEE ALSO

`gui_hide_palette(2)`
`gui_show_palette(2)`

gui_create_tk_task_page

Create a Tk task page.

SYNTAX

```
gui_create_tk_task_page  
-name PageName  
-create_command TclProcName  
[-default]
```

PageName *String*
TclProcName *String*

ARGUMENTS

-name *PageName*

PageName specifies the name of the new task page. Task page names must be unique.

-create_command *TclProcName*

TclProcName specifies the name of Tcl procedure to call to construct the Tk widget.

DESCRIPTION

This command creates a task page which can be shown in the task assistant. The name option value is the unique name of the page which can be referenced from the `gui_create_task_item` command. The `create_command` option value is the Tcl procedure name which will be called to populate the page when a task item referencing this task page is selected by the user. The procedure is called with the name of the Tk widget to populate and the name of the parent widget.

EXAMPLES

The following simple example creates a new task, then defines a Tk widget task page, then creates task items referencing the task page.

```
proc create_widget_proc {widgetName parentName} {  
    #populate $widgetName with Tk  
}
```

```
gui_create_task -name MyAppTask -item_root MyAppTaskRoot
```

```
gui_create_tk_task_page -name InputFormPage -create_command create_widget_proc
```

```
gui_create_task_item -name "Application->Input Form" -task MyAppTask -page InputFormPage
```

SEE ALSO

`gui_create_task(2)`

`gui_create_task_item(2)`

`gui_create_html_task_page(2)`

gui_create_toolbar

Create a toolbar with the specified name.

SYNTAX

```
string gui_create_toolbar -name ToolBarName
    [-title Title]
    [-dock_side DockSide]
    [-hidden]
    [-window_type WindowTypeName]
```

ToolBarName *String*
Title *String*
DockSide *String*
WindowTypeName *String*

ARGUMENTS

-name *ToolBarName*

ToolBarName is the toolbar identifier for the new toolbar.

-title *Title*

Title specifies the title or caption of the toolbar.

-dock_side *DockSide*

DockSide specifies the side of the window the tool docks to by default. The legal values are top, bottom, left, right.

-hidden

-hidden specifies the toolbar should be hidden by default.

-window_type *WindowTypeName*

WindowTypeName specifies the type of top level window whose menu bar is to be modified. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable `gui_default_window_type(3)` specifies the default window type for the application.

DESCRIPTION

This command creates a toolbar with the given name and title. Returns the name of the toolbar created.

EXAMPLES

Create a toolbar called mytoolbar and add a button to it that invokes the File->Quit menu item.

```
gui_create_toolbar -name mytoolbar  
gui_create_toolbar_item -name mytoolbar -menu "File->Quit"
```

SEE ALSO

```
gui_set_hotkey(2)  
gui_report_hotkeys(2)  
gui_create_menu(2)  
gui_delete_menu(2)  
gui_delete_toolbar(2)  
gui_create_toolbar_item(2)  
gui_delete_toolbar_item(2)  
gui_get_toolbar_names(2)  
gui_default_window_type(3)  
gui_custom_setup_files(3)
```

gui_create_toolbar_item

Create a button in the specified toolbar.

SYNTAX

```
string gui_create_toolbar_item -toolbar ToolBarName  
    <-menu MenuName|-separator SeparatorName>  
    [-window_type WindowTypeName]
```

ToolBarName *String*
MenuName *String*
SeparatorName *String*
WindowTypeName *String*

ARGUMENTS

-toolbar *ToolBarName*

ToolBarName specifies the toolbar that the new toolbar item will be added to.

-menu *MenuName*

MenuName specifies the name of the menu item to be invoked from this toolbar button. The "name" of a menu is a string which specifies the hierarchy of menu labels connected with '>'. For example, File->Quit specifies the Quit menu item under the File menu. These names are allowed to also specify the mnemonic for the menu text which is specified with an embedded '&', but they are not required to do so. This option is mutually exclusive with the -separator option.

-separator *SeparatorName*

SeparatorName specifies the unique separator name within the toolbar. This option is mutually exclusive with the -menu option.

-window_type *WindowTypeName*

WindowTypeName specifies the type of top level window whose menu bar is to be modified. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable `gui_default_window_type(3)` specifies the default window type for the application.

DESCRIPTION

This command adds a toolbar button to the specified toolbar using information described in the -menu option. The -menu option specifies the hierarchical menu name for some menu item. Thus, the toolbar icon becomes another interface for a menu item. Returns the item name if successful.

EXAMPLES

Create a toolbar called mytoolbar and add a button to it that invokes the File->Quit menu item.

```
gui_create_toolbar -name mytoolbar  
gui_create_toolbar_item -name mytoolbar -menu "File->Quit"
```

SEE ALSO

- gui_set_hotkey(2)
- gui_report_hotkeys(2)
- gui_create_menu(2)
- gui_delete_menu(2)
- gui_create_toolbar(2)
- gui_delete_toolbar(2)
- gui_delete_toolbar_item(2)
- gui_get_toolbar_names(2)
- gui_default_window_type(3)
- gui_custom_setup_files(3)

gui_create_window

Creates a window of the specified window type.

SYNTAX

```
string gui_create_window  
-type window_type  
[ -parent parent_window_id ]  
[ -show_state window_state ]  
[ -title title ]  
[ -rect rect | -size {width height} ]  
[ -icon icon ]
```

Data Types

<i>window_type</i>	string
<i>parent_window_id</i>	string
<i>window_state</i>	string
<i>title</i>	string
<i>rect</i>	rectangle
<i>width</i>	integer
<i>height</i>	integer
<i>icon</i>	string

ARGUMENTS

-type *window_type*

Specifies the type of window you are creating. The tool creates an instance of this window type.

You can display a list of the supported window types by using the **gui_get_window_types** command.

-parent *parent_window_id*

Specifies the window ID of the parent top-level window for the view window you are creating.

If you do not specify a window ID and the window type is not a top-level window type, the tool uses the current top-level window. If there is no current top-level window, the tool creates a new top-level window automatically to host the window you are creating.

-show_state *create_window_state*

Specifies the window state in which to display the window you are creating. The result varies depending on whether the window is a top-level window or a view window.

For a top-level window, you can specify one of the following states:

maximized -- show maximized (fill whole screen)
minimized -- show iconified
normal -- show at preferred size

Note that the window manager might not fully honor the specified state. See your window manager documentation for more details.

For a view window, you can specify one of the following states:

maximized -- show maximized in view area and raise to top
minimized -- show iconified in view area
normal -- show at preferred size in view area and raise to top

If you display a view window in its maximized state, the tool also maximizes any existing view windows. If you display a view window in its minimized or normal state, the tool changes any maximized view windows to their normal states.

The default is `-show_state normal`.

-title *title*

Specifies the title for the top-level or view window you are creating.

Note that the window manager might not fully honor the specified top-level window title string. See your window manager documentation for more details.

-rect *rect*

Specifies the size and position of the window in the following format:

`{{x1 y1} {x2 y2}}`

The coordinates `{x1 y1}` specify the location of the top-left corner, and the coordinates `{x2 y2}` specify the location of the bottom-right corner. These coordinates are relative to the top-left corner of the screen, for a top-level window, or to the top-right corner of the view area in the parent top-level window, for a view window.

Note that the window manager might not fully honor the specified top-level window geometry. See your window manager documentation for more details.

The **-rect** option and the **-size** option are mutually exclusive.

-size {*width height*}

Specifies the width and height of the window.

Note that the window manager might not fully honor the specified top-level window geometry. See your window manager documentation for more details.

The **-size** option and the **-rect** option are mutually exclusive.

-icon *icon*

Specifies the image to be used for the top-level or view window icon.

For a view window, the icon appears in the top-left corner of the title bar when the window is in its normal or minimized state and at the left side of the menu bar when the window is maximized.

For a top-level window, the icon might not be displayed when the window is in its normal, minimized, or maximized state. See your window manager documentation for more details.

DESCRIPTION

This command creates a window of the specified type and returns the window ID. The window type that you specify controls whether the window is a top-level window or a view window.

EXAMPLES

The following example creates a top-level layout window and a minimized layout view window.

```
prompt> set top [gui_create_window -type LayoutWindow]
LayoutWindow.1
prompt> gui_create_window -type Layout -parent $top \
-show_state minimized
layout.1
```

SEE ALSO

- gui_close_window(2)
- gui_exist_window(2)
- gui_show_window(2)
- gui_get_window_types(2)
- gui_get_window_ids(2)
- gui_set_active_window(2)
- gui_get_current_window(2)

gui_delete_attrgroup

Removes a group of attributes or all attribute groups for an object type.

SYNTAX

```
status gui_delete_attrgroup  
-class design_object  
-name name  
[-all]
```

Data Types

<i>design_object</i>	string
<i>name</i>	string

ARGUMENTS

-class *design_object*

Specifies the type of design object.

-name *name*

Specifies the name of the attribute group to be removed for the specified object type.

-all

Removes all attribute groups defined for the specified object type.

DESCRIPTION

The **gui_delete_attrgroup** command removes the specified attribute group or all attribute groups for the specified object type.

EXAMPLES

```
prompt> gui_delete_attrgroup -class Cell -name "Hierarchy"
```

```
prompt> gui_delete_attrgroup -class Cell -all
```

SEE ALSO

`gui_create_attrgroup(2)`
`gui_list_attrgroups(2)`
`gui_update_attrgroup(2)`

gui_delete_menu

Delete a menu item for the toplevel menubar or a context menu

SYNTAX

```
string gui_delete_menu  
[-menu HierMenuName | -all]  
[-window_type WindowTypeName | -root ContextMenuRoot]
```

HierMenuName *String*
WindowTypeName *String*
ContextMenuRoot *String*

ARGUMENTS

-menu *HierMenuName*

HierMenuName specifies the hierarchical menu name of the menu item to be deleted. The "name" of a menu is a string which specifies the hierarchy of menu labels connected with '->'. For example, File->Quit specifies the Quit menu item under the File menu. These names are allowed to also specify the mnemonic for the menu text which is specified with an embedded '&', but they are not required to do so. This option is mutually exclusive with the -all option.

-all

This option flag specifies all menu items in a menu root to be deleted. The option cannot be given with the -window_type option.

-window_type *WindowTypeName*

WindowTypeName specifies the type of top level window whose menu bar is to be modified. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable gui_default_window_type(3) specifies the default window type for the application.

-root *ContextMenuRoot*

ContextMenuRoot specifies the context menu root to from which the menu item will be added. A context menu root name groups menu items to be included in a pop-up context menu. A context menu is usually brought up by right clicking on a window and is used to provide context-sensitive features. (Note the difference between a context menu root and the menu root use by a toplevel window's menu bar .)

DESCRIPTION

This command deletes a menu item for the menu bar of a top level window. The menu hierarchy separator is denoted by "->". If there

is a toolbar button or hotkeys defined for the menu being deleted they will also be deleted. If the path to the menu is a submenu that menu item and all its sub-items will be deleted. The option `-all` may be used with a menu root name to remove all menu items from a menu. This is useful if a user wishes to customize or modify an entire context menu. These settings are stored either in a user-specific setup file or one that is shared across the installation. The gui initialization sequence will load builtin system configuration, and then apply the installation specific customizations, and finally the user's customizations. The installation specific setup file is specified by the variable `gui_custom_setup_file` and will have a default value of `$::synopsys/admin/setup/synopsys_galileo_gui.tcl`. This can be overridden in the `.synopsys_dc.setup` file if it is desired. If the file specified by this variable exists then it will be loaded after the gui is initialized. The user setup file is `~/synopsys_galileo_gui.tcl`.

EXAMPLES

The following example deletes the Quit item from the File menu.

```
gui_delete_menu -menu "File->Quit"
```

SEE ALSO

- `gui_set_hotkey(2)`
- `gui_report_hotkeys(2)`
- `gui_create_menu(2)`
- `gui_create_toolbar(2)`
- `gui_delete_toolbar(2)`
- `gui_create_toolbar_item(2)`
- `gui_delete_toolbar_item(2)`
- `gui_default_window_type(3)`
- `gui_custom_setup_files(3)`

gui_delete_toolbar

Delete a toolbar with the specified name.

SYNTAX

```
string gui_delete_toolbar -name ToolBarName  
[-window_type WindowTypeName]
```

ToolBarName *String*
WindowTypeName *String*

ARGUMENTS

-name *ToolBarName*

ToolBarName is the toolbar identifier for the toolbar to be deleted.

-window_type *WindowTypeName*

WindowTypeName specifies the type of top level window whose menu bar is to be modified. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable `gui_default_window_type(3)` specifies the default window type for the application.

DESCRIPTION

This command deletes a toolbar and all of its toolbar items.

EXAMPLES

Delete a toolbar called mytoolbar.

```
gui_delete_toolbar -name mytoolbar
```

SEE ALSO

```
gui_set_hotkey(2)
gui_report_hotkeys(2)
gui_create_menu(2)
gui_delete_menu(2)
gui_create_toolbar(2)
gui_create_toolbar_item(2)
gui_delete_toolbar_item(2)
gui_get_toolbar_names(2)
gui_default_window_type(3)
gui_custom_setup_files(3)
```

gui_delete_toolbar_item

Remove a toolbar button specified by the menu name from the specified toolbar.

SYNTAX

```
string gui_delete_toolbar_item -toolbar ToolBarName  
    <-menu MenuName|-separator SeparatorName>  
    [-window_type WindowTypeName]
```

ToolBarName *String*
MenuName *String*
SeparatorName *String*
WindowTypeName *String*

ARGUMENTS

-toolbar *ToolBarName*

ToolBarName specifies the toolbar to be modified.

-menu *MenuName*

ItemName specifies the hierarchical menu name for some menu item, that this toolbar item is associated with. This option is mutually exclusive with the -separator option.

-separator *SeparatorName*

SeparatorName specifies the unique separator name within the toolbar. This option is mutually exclusive with the -item option.

-window_type *WindowTypeName*

WindowTypeName specifies the type of top level window whose menu bar is to be modified. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable `gui_default_window_type(3)` specifies the default window type for the application.

DESCRIPTION

This command deletes a toolbar item specified by *MenuName* or *SeparatorName* from the specified toolbar. Returns the name of the deleted toolbar item if successful.

EXAMPLES

Delete the button bound to File->Quit from the toolbar mytoolbar.

```
gui_delete_toolbar_item -name mytoolbar -menu "File->Quit"
```

SEE ALSO

- gui_set_hotkey(2)
- gui_report_hotkeys(2)
- gui_create_menu(2)
- gui_delete_menu(2)
- gui_create_toolbar(2)
- gui_delete_toolbar(2)
- gui_create_toolbar_item(2)
- gui_get_toolbar_names(2)
- gui_default_window_type(3)
- gui_custom_setup_files(3)

gui_eval_command

Executes and optionally logs a tool command language (Tcl) command.

SYNTAX

```
string gui_eval_command  
-command command  
[-echo]  
[-history]  
[-honor_preview | -preview]
```

Data Types

command string

ARGUMENTS

-command *command*

Specifies the Tcl command to execute and to record in the command replay log.

-echo

Echoes the command and displays the command result in the console log view. By default, the tool does not echo the command or display the result in the log view.

-history

Appends the command to the command history list in the console history view. By default, the tool does not append the command to the command history list.

-honor_preview

This option is mutually exclusive with the -preview option. Honors the Preview Command Only setting in the dialog. By default, the tool does not honor the Preview Command Only setting.

-preview

This option is mutually exclusive with the -honor_preview option. Runs the given command in preview mode, irregardless of the current preview mode set from a command dialog. When -preview is given, -echo is turned on since the previewed command cannot be seen in the xterm if -echo is turned off. As with all commands issued from command dialogs when in preview mode, if a script editor is present, then the previewed command is redirected to the script editor. Using this option does not set or clear the global preview mode in the application.

DESCRIPTION

This command executes a Tcl command and records it in the command replay log. Use the **-echo** option to echo the command and its result in the console log view. Use the **-history** option to append the command to the command history list. Use the **-honor_preview** option to honor the Preview Command Only setting. Use the **-preview** option to preview the command only without executing it.

The result of a nested command is passed back as the result of **gui_eval_command**.

EXAMPLES

The following example evaluates the command "echo abc".

```
prompt> gui_eval_command -command {echo abc}  
abc
```

The following example previews the command "echo abc".

```
prompt> gui_eval_command -command {echo abc} -preview  
abc
```

gui_execute_menu_item

Execute a menu item in the currently active window.

SYNTAX

```
status gui_execute_menu_item
{ -menu menu_item_id }
```

Data Types

menu_item_id string

ARGUMENTS

-menu *menu_item_id*

menu_item_id specifies the hierarchical name of the menu item to execute. The "name" of a menu is a string which specifies the hierarchy of menu labels connected with '->'. For example, File->Quit specifies the Quit menu item in the File menu. These names are allowed to also specify the keyboard accelerator for the menu text which is specified with an embedded '&', but they are not required to do so.

DESCRIPTION

This command checks for the existence and enabled state of the given menu item in the currently active window. If it exists and is enabled, the menu item is executed.

The command returns and result value string from the executed menu item, if any.

EXAMPLES

The following example activates the main window named Toplevel.2, then executes the **Hierarchy Browser** menu item in the **View** menu in Toplevel.2.

```
shell> gui_set_active_window -window Toplevel.2
shell> gui_execute_menu_item -menu "View->Hierarchy Browser"
```

SEE ALSO

`gui_create_menu(2)`
`gui_set_active_window(2)`

gui_exist_pref_category

Check the existence of a preference category.

SYNTAX

```
bool gui_exist_pref_category  
-category Category
```

ARGUMENTS

-category *Category*

Category specifies the category.

DESCRIPTION

This command checks the existence of a preference category. Return "1" if the specified category exists, otherwise return "0".

SEE ALSO

```
gui_create_pref_category(2)  
gui_create_pref_key(2)  
gui_set_pref_value(2)  
gui_get_pref_value(2)  
gui_remove_pref_key(2)  
gui_get_pref_keys(2)  
gui_exist_pref_key(2)  
gui_update_pref_file(2)
```

gui_exist_pref_key

Check the existence of a preference key.

SYNTAX

```
bool gui_exist_pref_key  
  -key Key  
  [-category Category]
```

ARGUMENTS

-key *Key*

Key specifies the preference key of interest.

-category *Category*

Category specifies the category. Default category will be used if missing.

DESCRIPTION

This command checks the existence of a preference key given the key name and the preference category. Return "1" if the specified key exists, otherwise return "0".

SEE ALSO

```
gui_create_pref_category(2)  
gui_create_pref_key(2)  
gui_set_pref_value(2)  
gui_get_pref_value(2)  
gui_remove_pref_key(2)  
gui_get_pref_keys(2)  
gui_exist_pref_category(2)  
gui_update_pref_file(2)
```

gui_exist_window

Check for the existence of specified window or window type

SYNTAX

```
status gui_exist_window
{ -window window_id | -type window_type }
[ -parent window_id ]
```

Data Types

```
window_id    string
window_type  string
```

ARGUMENTS

-window *window_id*

Specifies the window id to test existence for.

This option is mutually exclusive with the *-type* option.

-type *window_type*

Specifies that the existence of a window of this window type is tested for.

If the *-parent* option is specified then the existence of a window of this window type is tested for in the specified toplevel window.

If the *-parent* option is **not** specified then the existence of a window of this window type is tested for in any toplevel window.

This option is mutually exclusive with the *-window* option.

-parent *window_id*

Specifies the toplevel window to check for existence in.

Note: This option is only relevant if the *-type* option is specified.

DESCRIPTION

This command checks the existence of the specified window, or window type across toplevel windows or within a specified toplevel window.

The command returns "1" or "0" for success or failure respectively.

EXAMPLES

The following example will create a toplevel window and a child layout view window, then check the existence of the command layout window.

```
shell> set top [gui_create_window -type TopLevel]
shell> set x [gui_create_window -type Layout -parent $top]
shell> set layout_exist [gui_exist_window $x]
shell> if { $layout_exist == 1 } {echo "layout exist" }
```

SEE ALSO

- gui_close_window(2)
- gui_create_window(2)
- gui_show_window(2)
- gui_get_window_types(2)
- gui_get_window_ids(2)
- gui_set_active_window(2)
- gui_get_current_window(2)

gui_get_annotations

Return a collection of layout annotations

SYNTAX

```
string gui_get_annotations  
  [-window window_name]  
  [-group group_id]  
  [-within region]  
  [-intersect]  
  [-at location]  
  [-filter filter]  
  [-nocase]  
  [-regexp]
```

```
string window_name  
string group_name  
string filter  
rectangle region  
location point
```

ARGUMENTS

-window *window_name*

Specifies the instance name of the layout window. If not specified implies all windows.

-group *group_name*

Specifies the annotation group name. If not specified implies all groups.

-within *region*

Search for annotations contained within the specified rectangle.

-intersect

Use this option with the -within option. When -intersect is specified, all annotations that touch the search rectangle are also returned.

-at *location*

Search for annotations that are close to the specified point.

-filter *filter*

Only return annotations that satisfy the filter expression.

-nocase

Ignore case in the filter expression. Option requires -filter.

-regexp

Use regular expressions in the filter expression. Option requires -filter.

DESCRIPTION

Returns a collection of `gui_annotation` objects that satisfy the requirements.

EXAMPLES

The following example retrieves all the annotations in the Layout.1 window.

```
shell> gui_get_annotations -window Layout.1
```

This example shows how to use the `client_data` attribute to find annotations with that setting.

```
shell> gui_get_annotations -filter client_data==MyData
```

This example shows how to find annotations that are within a specified rectangle

```
shell> gui_get_annotations -within {{10.5 2.0} {23.5 14.6}}
```

To also find the annotations that touch the search rectangle:

```
shell> gui_get_annotations -within {{10.5 2.0} {23.5 14.6}} -intersect
```

SEE ALSO

`gui_add_annotation` **`gui_remove_annotations`**

gui_get_bucket_option

Returns the value of an option for a bucket in the specified visual mode or map mode.

SYNTAX

```
string gui_get_bucket_option  
-map map_name  
-bucket bucket_name  
-option option_name  
[-default]
```

Data Types

<i>map_name</i>	string
<i>bucket_name</i>	string
<i>option_name</i>	string

ARGUMENTS

-map *map_name*

Specifies the name of the visual mode or map mode.

-bucket *bucket_name*

Specifies the name of the bucket.

-option *option_name*

Specifies the name of the option whose value is to be returned.

-default

Returns the default value of the option instead of the current value.

DESCRIPTION

This command returns the current or default value of an option for a bucket in a visual mode or map mode. You must specify the visual mode or map mode name, the bucket name, and the option name. Use **-default** if you want the command to return the default option value instead of the current option value.

EXAMPLES

The following example returns the current color value for *bucket1* in the global route congestion map:

```
prompt> gui_get_bucket_option -map AREAPARTITION \  
       -bucket bucket1 -option color
```

SEE ALSO

- `gui_get_bucket_option_list(2)`
- `gui_get_map_list(2)`
- `gui_get_map_option(2)`
- `gui_get_map_option_list(2)`
- `gui_set_bucket_option(2)`
- `gui_set_map_option(2)`

gui_get_bucket_option_list

Lists the available options for buckets in the specified visual mode or map mode.

SYNTAX

```
string gui_get_bucket_option_list  
-map map_name
```

Data Types

map_name string

ARGUMENTS

-map *map_name*

Specifies the name of the visual mode or map mode.

DESCRIPTION

This command displays a list of all options that are available for the buckets in the specified visual mode or map mode.

EXAMPLES

The following example displays a list of the available options for buckets in the global route congestion map:

```
prompt> gui_get_bucket_option_list -map AREAPARTITION
```

SEE ALSO

```
gui_get_bucket_option(2)  
gui_get_map_list(2)  
gui_get_map_option(2)  
gui_get_map_option_list(2)
```

```
gui_set_bucket_option(2)
gui_set_map_option(2)
```

gui_get_cell_block_marks

Gets a list of the block mark string values on one or more cells.

SYNTAX

```
list gui_get_cell_block_marks
    cells
```

Data Types

cells list

ARGUMENTS

cells

Lists one or more cell name patterns or cell collections.

DESCRIPTION

This command gets the block mark string values from one or more hierarchical or leaf-level cell instances.

EXAMPLES

The following example sets the block mark for a hierarchical cell instance identified by a cell name, and then gets the block mark from that cell instance:

```
prompt> gui_set_cell_block_marks I_TOP/I_ALU ALU
prompt> gui_get_cell_block_marks I_TOP/I_ALU
ALU
```

The following example sets the block mark for a hierarchical cell instance identified by a nested run of the **get_cells** command, and then gets the block mark from that cell instance:

```
prompt> gui_set_cell_block_marks [get_cells I_TOP/I_ALU] ALU
prompt> gui_get_cell_block_marks [get_cells I_TOP/I_ALU]
ALU
```


The following example shows how using the **get_attribute** command to query the **block_mark** attribute is an alternative to using the **gui_get_cell_block_marks** command:

```
prompt> gui_set_cell_block_marks I_TOP/I_ALU ALU
prompt> gui_get_cell_block_marks I_TOP/I_ALU
ALU
prompt> get_attribute -class cell I_TOP/I_ALU block_mark
ALU
```

SEE ALSO

[gui_set_cell_block_marks\(2\)](#)
[gui_remove_cell_block_marks\(2\)](#)

gui_get_current_task

Get the name of the current task.

SYNTAX

string **gui_get_current_task**

DESCRIPTION

The `gui_get_current_task` command queries the name of the current task.

SEE ALSO

`gui_create_task(2)`
`gui_get_current_task_item(2)`
`gui_get_current_task_page(2)`
`gui_set_current_task(2)`
`gui_get_task_list(2)`

gui_get_current_task_item

Get the name of the current task item.

SYNTAX

string **gui_get_current_task_item**

DESCRIPTION

The `gui_get_current_task_item` command queries the name of the current task item.

SEE ALSO

`gui_get_current_task(2)`
`gui_get_current_task_page(2)`
`gui_set_current_task(2)`
`gui_get_task_list(2)`

gui_get_current_task_page

Get the name of the current task page.

SYNTAX

string **gui_get_current_task_page**

DESCRIPTION

The `gui_get_current_task_page` command queries the name of the current task page.

SEE ALSO

`gui_create_task(2)`
`gui_get_current_task(2)`
`gui_get_current_task_item(2)`
`gui_set_current_task(2)`
`gui_get_task_list(2)`

gui_get_current_window

Retrieves the window ID of the active top-level or view window.

SYNTAX

```
string gui_get_current_window  
  [-toplevel | -view | -parent parent_window_id]  
  [-hier_name ]  
  [-types window_type_list]  
  [-mru ]
```

Data Types

parent_window_id string
window_type_list list

ARGUMENTS

-toplevel

Retrieves the active top-level window unless you also specify the **-types** option.

If you also specify the **-types** option, the **-toplevel** option retrieves the most recently active top-level window that has one of the specified window types.

The **-toplevel**, **-view**, and **-parent** options are all mutually exclusive.

-view

Retrieves the active view window in the active top-level window unless you also specify the **-types** option, the **-mru** option, or both.

if you also specify the **-types** option but not the **-mru** option, the **-view** option retrieves the most recently active view window with one of the specified window types in the active top-level window.

if you also specify the **-mru** option but not the **-types** option, the **-view** option retrieves the most recently active view window in any of the open top-level windows.

If you also specify both the **-types** option and the **-mru** option, the **-view** option retrieves the most recently active view window that has one of the specified window types and is in an open top-level window.

The **-view**, **-toplevel**, and **-parent** options are all mutually exclusive.

-parent *parent_window_id*

Retrieves the active view window in the specified top-level window unless you also specify the **-types** option.

If you also specify the **-types** option, the **-parent** option retrieves the most recently active view window that has any of the specified window types and is in the specified top-level window.

The **-parent**, **-toplevel**, and **-view** options are all mutually exclusive.

-hier_name

Returns the window ID in Qtcl format, which is similar to a full path name for the window, for use with the `qtcl_*` commands.

Note that a value returned in this format is not usable by `gui_*` commands.

-types window_type_list

Forces the tool to retrieve a window with one of the specified window types.

Use this option to restrict the search to certain types of windows. If you specify multiple types, they should be consistent (all `toplevel` window types or all view window types), but nonconsistency is not considered an error. The first type in the list determines whether the tool retrieves a top-level window or a view window when the **-toplevel** and **-view** options are not specified.

When you use the **-types** option, you should not use the **-toplevel** option or the **-view** option because the tool can determine from the list whether the window should be a top-level window or a view window. If you specify one of these options, any types that do not correspond to the option are silently ignored. For example, if you specify **-view**, then all top-level types are ignored.

-mru

Forces the tool to retrieve the most recently active view window from among all the open top-level and view windows. This option is effective only when you specify the **-view** option or when all the window types you specify with the **-types** option are view window types.

DESCRIPTION

This command retrieves the window ID of the active top-level or view window based on the options that you specify, or returns an empty string if no match is found.

A view window is a child window of a top-level window and is created with the **gui_create_window** command.

If you do not specify the **-toplevel**, **-view**, and **-parent** options, the tool uses the **-types** option to determine the window type. If you also do not specify the **-types** option, the tool uses the **-toplevel** option by default.

EXAMPLES

The following example retrieves the active top-level window:

```
prompt> gui_get_current_window -toplevel
```

The following example retrieves the active view window:

```
prompt> gui_get_current_window -view
```

The following example retrieves the most recently active layout view:

```
prompt> gui_get_current_window -types "Layout" -mru
```

The following example return \$cons:

```
prompt> set top1 [gui_create_window -type LayoutWindow]
prompt> set top2 [gui_create_window -type LayoutWindow]
prompt> set cons [gui_create_window -type Layout -parent $top]
prompt> gui_get_current_window -parent $top
```

SEE ALSO

- gui_close_window(2)
- gui_exist_window(2)
- gui_show_window(2)
- gui_get_window_types(2)
- gui_get_window_ids(2)
- gui_set_active_window(2)
- gui_create_window(2)

gui_get_highlight

Get a collection of highlighted objects.

SYNTAX

```
string gui_get_highlight  
  [-color color_id | -all_colors]  
  [-return_select_bus]
```

Data Types

color_id string

ARGUMENTS

-color *color_id*

Specifies the color of objects to be returned. If neither -color nor -all_colors is used, the current color is assumed. The allowed colors are yellow, orange, red, green, blue, purple, light_orange, light_red, light_green, light_blue, and light_purple.

-all_colors

Specifies that all highlighted objects are to be returned.

-return_select_bus

Create a new selection bus in which to store the result instead of returning a collection of objects.

DESCRIPTION

The gui_get_highlight command returns a collection of objects highlighted with the specified colors.

EXAMPLES

Get a collection of green highlighted objects.

```
prompt> gui_get_highlight -color green
```


Get a collection of all highlighted objects.

```
prompt> gui_get_highlight -all_colors
```

SEE ALSO

`gui_change_highlight(2)`
`gui_get_highlight_options(2)`
`gui_set_highlight_options(2)`

gui_get_highlight_options

Query the options that control highlighting.

SYNTAX

```
string gui_get_highlight_options  
[-current_color | -all_colors | -auto_cycle_color]
```

ARGUMENTS

-current_color

This option returns a string which is the current highlight color, or the default color for highlighting operations.

-all_colors

This option returns Tcl list of all of the colors which are allowed to be used for highlighting.

-auto_cycle_color

This option returns the current value of the auto cycle setting ("true" or "false").

DESCRIPTION

The `gui_get_highlight_options` command queries the settings which control highlighting. Exactly one option must be provided.

EXAMPLES

Get the current highlight color.

```
prompt> gui_get_highlight_options -current_color
```

Get all highlight colors.

```
prompt> gui_get_highlight_options -all_colors
```

SEE ALSO

`gui_change_highlight(2)`
`gui_get_highlight(2)`
`gui_set_highlight_options(2)`

gui_get_map_list

Lists the current visual and map modes.

SYNTAX

string **gui_get_map_list**

ARGUMENTS

The **gui_get_map_list** command has no arguments.

DESCRIPTION

The **gui_get_map_list** command returns a list of existing visual and map modes.

EXAMPLES

The following example gets the names of all available visual and map modes:

```
prompt> gui_get_map_list
```

SEE ALSO

- gui_get_bucket_option(2)
- gui_get_bucket_option_list(2)
- gui_get_map_option(2)
- gui_get_map_option_list(2)
- gui_set_bucket_option(2)
- gui_set_map_option(2)
- gui_show_map(2)

gui_get_map_option

Returns the value of an option for the specified visual mode or map mode.

SYNTAX

```
string gui_get_map_option  
-map map_name  
-option option_name  
[-default]
```

Data Types

<i>map_name</i>	string
<i>option_name</i>	string

ARGUMENTS

-map *map_name*

Specifies the name of the visual mode or map mode.

-option *option_name*

Specifies the name of the option whose value is to be returned.

-default

Returns the default value of the option instead of the current value.

DESCRIPTION

This command returns the current or default value of an option for a visual mode or map mode. You must specify the visual mode or map mode name and the option name. Use **-default** if you want the command to return the default option value instead of the current option value.

EXAMPLES

The following example returns the current number of buckets for the global route congestion map:

```
prompt> gui_get_map_option -map AREAPARTITION \  
       -option num_buckets
```

SEE ALSO

- `gui_get_bucket_option(2)`
- `gui_get_bucket_option_list(2)`
- `gui_get_map_list(2)`
- `gui_get_map_option_list(2)`
- `gui_set_bucket_option(2)`
- `gui_set_map_option(2)`

gui_get_map_option_list

Lists the available options for the specified visual mode or map mode.

SYNTAX

```
string gui_get_map_option_list  
-map map_name
```

Data Types

map_name string

ARGUMENTS

-map *map_name*

Specifies the name of the visual mode or map mode.

DESCRIPTION

This command displays a list of all options that are available for the specified visual mode or map mode.

EXAMPLES

The following example displays a list of the available options for the global route congestion map:

```
prompt> gui_get_map_option_list -map AREAPARTITION
```

SEE ALSO

```
gui_get_bucket_option(2)  
gui_get_bucket_option_list(2)  
gui_get_map_list(2)  
gui_get_map_option(2)
```

```
gui_set_bucket_option(2)
gui_set_map_option(2)
```

gui_get_menu_roots

Return a sorted list of all menu roots used by the application.

DESCRIPTION

This command returns a list of all the menu roots used by the application. The menu roots include menu roots used by toplevel menu bars and menu roots used by context menus. A menu root is used to group menu items that share the same menu root together. All items under each toplevel window's menu bar shares the same menu root; similarly, all items in a context menu (menu brought up with right mouse click in a window) shares the same context menu root.

SEE ALSO

`gui_create_menu(2)`

gui_get_mouse_tool_option

Get the value of a mouse tool option

SYNTAX

```
string gui_get_mouse_tool_option -tool string  
-option string
```

```
string string  
string string
```

ARGUMENTS

-tool *string*

Tool to get option for.

-option *string*

Option to get.

DESCRIPTION

This command returns value of a mouse tool option .

EXAMPLES

```
> gui_get_mouse_tool_option -tool SELECT -option InputMode  
Smart
```

SEE ALSO

```
gui_mouse_tool(2)  
gui_set_mouse_tool_option(2)
```

gui_get_pref_keys

Return a list of preference keys under the specified category.

SYNTAX

```
string gui_get_pref_keys  
-category Category
```

ARGUMENTS

-category *Category*

Category specifies the category to use.

DESCRIPTION

Return a list of preference keys under the specified category.

SEE ALSO

```
gui_create_pref_category(2)  
gui_create_pref_key(2)  
gui_set_pref_value(2)  
gui_get_pref_value(2)  
gui_remove_pref_key(2)  
gui_exist_pref_category(2)  
gui_exist_pref_key(2)  
gui_update_pref_file(2)
```

gui_get_pref_value

Get the value of a preference key.

SYNTAX

```
string gui_get_pref_value  
-key Key [-category Category]
```

ARGUMENTS

-key *Key*

Key specifies the key to retrieve the value from.

-category *Category*

Category specifies the category to search for the key. If not specified, a default category will be used.

DESCRIPTION

This command retrieves the value of a preference key from the specified key name and category.

EXAMPLES

The following example create a user-defined category *some_category* for storing preferences and then create a boolean key *some_key* under that category with the initial value of false. The command **gui_get_pref_value** is then used to retrieve the value of the preference key just created.

```
gui_create_pref_category -category some_category  
gui_create_pref_key -category my_category -key some_key -value_type bool -value false  
set x [gui_get_pref_value -category my_category -key some_key]
```

SEE ALSO

```
gui_create_pref_category(2)
gui_create_pref_key(2)
gui_set_pref_value(2)
gui_remove_pref_key(2)
gui_get_pref_keys(2)
gui_exist_pref_category(2)
gui_exist_pref_key(2)
gui_update_pref_file(2)
```

gui_get_region

Returns the coordinates of the current region rectangle or rectilinear polygon.

SYNTAX

list **gui_get_region**

DESCRIPTION

This command returns the coordinates of the rectangle or rectilinear polygon that defines the current region in layout region mode.

Note that the preferred command for typical extension writing is usually **gui_set_layout_user_command** instead of **gui_get_region** because **gui_set_layout_user_command** has snapping, mouse up drag, cancel, and user prompt capabilities.

EXAMPLES

```
prompt> gui_get_region
{{-1743.945 18.357} {42.833 1523.657}}
```

SEE ALSO

gui_set_region(2)
gui_set_layout_user_command(2)

gui_get_setting

Gets a setting on the specified window.

SYNTAX

```
string gui_get_setting  
-window WindowID  
{-setting Setting | -list}
```

ARGUMENTS

-window *WindowID*

WindowID specifies the window to get the setting

-setting *Setting*

Setting specifies the setting to get

-list

Specifies that the list of available settings needs to be returned

DESCRIPTION

Gets the list of available settings on the specified window. Windows are views or top level windows. The setting is a property of the window. The returned value is a string which has the type of the setting being read. Some windows might not have this function implemented.

EXAMPLES

```
prompt> gui_get_setting -window Layout.1 -setting showCell  
1
```

```
prompt> gui_get_setting -window Layout.1 -setting viewLevel  
2
```

SEE ALSO

`gui_set_setting(2)`

gui_get_task_list

Lists all the available task names.

SYNTAX

string **gui_get_task_list**

DESCRIPTION

The `gui_get_task_list` command queries the names of all available tasks.

DESCRIPTION

prompt> **gui_get_task_list** all {Block Implementation} {Design Planning}

SEE ALSO

`gui_create_task(2)`
`gui_set_task_list(2)`
`gui_set_current_task(2)`
`gui_get_current_task(2)`

gui_get_task_page

Return the name of the task page for the given task item.

SYNTAX

string **gui_get_task_page**

SYNTAX

gui_get_task_page
-task *TaskName*
-item *ItemName*

TaskName *String*
ItemName *String*

ARGUMENTS

-task *TaskName*

TaskName specifies the name of the task.

-item *ItemName*

ItemName specifies the hierarchical name of the task item for which to get the page name",

DESCRIPTION

This command returns the name of the task page for the task item identified by the option values. This is useful when you want to create a new task item re-using a task page that is invoked by an existing task item.

EXAMPLES

To reuse the task page from the built-in task item "Pin Assignment->Pin Placement" in the task "Hierarchical Design" in your own task flow, you can do the following:

```
prompt> set pageName [gui_get_task_page -task "Hierarchical Design"
                    -item "Pin Assignment->Pin Placement"]
prompt> gui_create_task_item -task myTask -name myItemName
                    -page $pageName
```

SEE ALSO

```
gui_create_task_item(2)
gui_set_current_task(2)
gui_get_task_list(2)
```

gui_get_toolbar_names

Return a Tcl list of the names of all the toolbars that have been created with the **gui_create_toolbar** command.

SYNTAX

```
void gui_get_toolbar_names  
[-window WindowId]
```

WindowId *String*

ARGUMENTS

-window *WindowId*

Return toolbars defined in this window. If omitted, return toolbars defined in the active window.

DESCRIPTION

Return a Tcl list of the names of all the toolbars that have been created with **gui_create_toolbar** command for the specified window.

SEE ALSO

gui_create_toolbar(2)
gui_create_toolbar_item(2)
gui_delete_toolbar(2)
gui_delete_toolbar_item(2)
gui_hide_toolbar(2)
gui_show_toolbar(2)

gui_get_window_ids

Get a list of window ids

SYNTAX

```
ids gui_get_window_ids
    [-parent window_id]
    [-type window_type]
```

Data Types

window_id string
window_type string

ARGUMENTS

-parent *window_id*

Specifies the toplevel window id to get child view window ids from.

-type *window_type*

Specifies that the windows id returned should be restricted to those of the specified type.

Note: as a type can be only be associated with either a toplevel window or a child view window (not both) the type implies whether toplevel or child view window ids are returned.

If this option is not specified then either all toplevel window ids are returned (*-parent* option not specified) or all child view window ids of a specified toplevel window id are returned (*-parent* option specified).

RETURNS

ids

Returned list of window ids.

DESCRIPTION

This command gets a list of the ids of all toplevel windows, all toplevel windows of a specified type, all child view windows of a specified type, or all child view windows of a specified type in a specified toplevel window.

EXAMPLES

The following example will create a toplevel window and a child layout window. Then this command will be used to retrieve the ids of the existing windows.

```
\prompt> set top [gui_create_window -type TopLevel]
\prompt> set cons [gui_create_window -type Layout -parent $top]
\prompt> set ids [gui_get_window_ids] # return list containing values of $top and value of $cons.
\prompt> set childs [gui_get_window_ids -parent $top] # return value of $cons.
\prompt> set layout_instances [gui_get_window_ids -type Layout] # return value of $cons.
```

SEE ALSO

- gui_close_window(2)
- gui_exist_window(2)
- gui_show_window(2)
- gui_get_window_types(2)
- gui_create_window(2)
- gui_set_active_window(2)
- gui_get_current_window(2)

gui_get_window_pref_categories

Get list of preference categories for object specified by window

SYNTAX

```
categories gui_get_window_pref_categories
  {-window window_id
  | -window_type window_type}
```

Data Types

```
window_id    string
window_type  string
```

ARGUMENTS

-window *window_id*

Specifies the name of the window that the preference categories will be retrieved from. This option is mutually exclusive with the -window_type option.

-window_type *window_type*

Specifies the name of the window type that instances of the type will have the preference categories be retrieved from. This option is mutually exclusive with the -window option.

RETURNS

categories

The returned preference categories.

DESCRIPTION

This command retrieves the list of preference categories for a window or window type.

EXAMPLES

The following example gets the preference categories for the TopLevel.1 window.

```
\prompt> ser cats [gui_get_window_pref_categories -window TopLevel.1]
```

SEE ALSO

- gui_set_window_pref_key(2)
- gui_get_window_pref_value(2)
- gui_get_window_pref_keys(2)
- gui_create_pref_category(2)
- gui_set_pref_value(2)
- gui_get_pref_value(2)
- gui_remove_pref_key(2)
- gui_get_pref_keys(2)
- gui_exist_pref_category(2)
- gui_exist_pref_key(2)
- gui_update_pref_file(2)

gui_get_window_pref_keys

Get list of preference categories for object specified by window

SYNTAX

```
categories gui_get_window_pref_keys
  {-window window_id
  | -window_type window_type}
  [-category category]
```

Data Types

```
window_id    string
window_type  string
category     string
```

ARGUMENTS

-window *window_id*

Specifies the name of the window that the preference categories will be retrieved from. This option is mutually exclusive with the -window_type option.

-window_type *window_type*

Specifies the name of the window type that instances of the type will have the preference categories be retrieved from. This option is mutually exclusive with the -window option.

-category *category*

Specifies the category that the key belongs to. If not specified, a default category will be assigned.

RETURNS

categories

The returned preference categories.

DESCRIPTION

This command retrieves the list of preference categories for a window or window type.

EXAMPLES

The following example gets the preference categories for the TopLevel.1 window.

```
\prompt> ser cats [gui_get_window_pref_keys -window TopLevel.1]
```

SEE ALSO

- gui_set_window_pref_key(2)
- gui_get_window_pref_value(2)
- gui_get_window_pref_categories(2)
- gui_create_pref_category(2)
- gui_set_pref_value(2)
- gui_get_pref_value(2)
- gui_remove_pref_key(2)
- gui_get_pref_keys(2)
- gui_exist_pref_category(2)
- gui_exist_pref_key(2)
- gui_update_pref_file(2)

gui_get_window_pref_value

Get preference value for object specified by window or window type

SYNTAX

```
value gui_get_window_pref_value
  {-window window_id
   | -window_type window_type}
  [-category category]
  -key key
```

Data Types

```
window_id    string
window_type  string
category     string
key          string
```

ARGUMENTS

-window *window_id*

Specifies the name of the window that the preference will be retrieved from. This option is mutually exclusive with the -window_type option.

-window_type *window_type*

Specifies the name of the window type that instances of the type will have the preference be retrieved from. This option is mutually exclusive with the -window option.

-category *category*

Specifies the category to search for the key. If not specified, a default category will be used.

-key *key*

Specifies the key to retrieve the value from.

RETURNS

value

The value of the key.

DESCRIPTION

This command retrieves the value of a preference key from the specified key name and category of the specified window or window type. If a preference is set on a window type, then all instances of that type will inherit the preference, so the value can be set on a type and retrieved from any instance of that type (or derived from that type).

EXAMPLES

The following example creates a preference for a window and then uses the **gui_get_window_pref_value** command to retrieve the value for the preference.

```
\prompt> gui_set_window_pref_key -window Console.1 -key test -value_type integer -value 201
\prompt> set x [gui_get_window_pref_value -window Console.1 -key test]
\prompt> if {$x == 201} { echo "success" }
```

SEE ALSO

- gui_set_window_pref_key(2)
- gui_get_window_pref_categories(2)
- gui_get_window_pref_keys(2)
- gui_create_pref_category(2)
- gui_set_pref_value(2)
- gui_get_pref_value(2)
- gui_remove_pref_key(2)
- gui_get_pref_keys(2)
- gui_exist_pref_category(2)
- gui_exist_pref_key(2)
- gui_update_pref_file(2)

gui_get_window_types

Get a list of window types

SYNTAX

```
types gui_get_window_types  
[-type token]
```

Data Types

token string

ARGUMENTS

-type *token*

Specifies whether to return toplevel windows ('toplevel') or child view windows ('child').

If toplevel window types are specified then the first type is guaranteed to be the default toplevel window type.

RETURNS

types

Returned list of window types.

DESCRIPTION

This command returns a list of existing window types. Instances of these types can be instantiated with the **gui_create_window** command. The returned list can be restricted to just toplevel windows or child view windows by specifying the *-type* option.

EXAMPLES

The following examples illustrate some usages of the command.

```
\prompt> gui_get_window_types  
\prompt> gui_get_window_types -type toplevel  
\prompt> gui_get_window_types -type child
```

SEE ALSO

- gui_close_window(2)
- gui_exist_window(2)
- gui_show_window(2)
- gui_create_window(2)
- gui_get_window_ids(2)
- gui_set_active_window(2)
- gui_get_current_window(2)

gui_hide_palette

Hides the specified palette.

SYNTAX

```
status gui_hide_palette
{ -name palette_id | -type palette_type }
[ -parent window_id ]
```

Data Types

```
palette_id  string
palette_type string
window_id   string
```

ARGUMENTS

-name *palette_id*

Specifies the palette name id.

Any matching palette of this name will be hidden.

This option precludes the *-type* option

-type *palette_type*

Specifies the palette type id. Any matching palette of this type will be hidden.

If the *-parent* option is specified, only palettes in the specified parent toplevel window are hidden. If the *-parent* option is not specified, all palettes of this type in all toplevel windows are hidden.

This option precludes the *-name* option

-parent *window_id*

Specifies the parent toplevel window to hide the palette type in.

Note: this option is only applicable if the *-type* option is specified.

DESCRIPTION

This command hides the specified palettes.

EXAMPLES

The following example hides all Layer palettes:

```
shell> gui_hide_palette -type Layout
```

SEE ALSO

[gui_show_palette\(2\)](#)

gui_hide_toolbar

Hides the specified toolbar or all the toolbars in the specified window.

SYNTAX

```
void gui_hide_toolbar  
  -toolbar tool_bar_name | -all  
  [-window window_id]
```

Data Types

```
tool_bar_name  string  
window_id      string
```

ARGUMENTS

-toolbar *tool_bar_name*

Specifies the toolbar you want to hide. The **-toolbar** option and the **-all** option are mutually exclusive.

-all

Hides all the toolbars in the specified window. The **-all** option and the **-toolbar** option are mutually exclusive.

-window *window_id*

Specifies the window in which you want to hide the specified toolbar or all toolbars. By default, the tool hides the toolbar or toolbars in the active window.

DESCRIPTION

This command hides either all the toolbars or the toolbar with the specified name in the window with the specified window ID.

EXAMPLES

The following example hides the File toolbar in the active window:

```
prompt> gui_hide_toolbar -toolbar File
```

The following example hides all the toolbars in the layout window named LayoutWindow.1:

```
prompt> gui_hide_toolbar -all -window LayoutWindow.1
```

SEE ALSO

- `gui_create_toolbar(2)`
- `gui_delete_toolbar(2)`
- `gui_create_toolbar_item(2)`
- `gui_delete_toolbar_item(2)`
- `gui_show_toolbar(2)`
- `gui_get_toolbar_names(2)`

gui_inspect_violations

Displays the specified DFT unified DRC violations in a new violation inspector window.

SYNTAX

```
status gui_inspect_violations  
[-type violation_type]  
violation_list
```

Data Types

```
violation_type  string  
violation_list  list
```

ARGUMENTS

-type *violation_type*

Specifies the type of violations (for example, D1) to display when you inspect multiple violations of the same type. This option affects how the command interprets the *violation_list* argument.

violation_list

Specifies the name of the violation instance to inspect in the violation inspector window. Violation instance names have the following format, which is similar to the format used in TetraMAX:

ViolationType-ViolationId

For example, violation 1 of type D1 is named D1-1.

If you include the **-type** option, *violation_list* specifies a list of one or more violation IDs. A violation ID is the violation number shown in the first column of violation browser.

DESCRIPTION

This command displays the specified DFT unified DRC violations in a new violation inspector window or in an existing violation inspector window that has been marked for reuse.

If no violation inspector window exists, the tool opens a new violation inspector window in a new top-level window. Subsequent violation inspector windows are opened in the active top-level window. New violation inspector windows are not marked for reuse.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command displays violations 5, 9, and 13 of type D1.

```
prompt> gui_inspect_violations -type D1 {5 9 13}
```

The following two commands display violation 4 of type D2.

```
prompt> gui_inspect_violations -type D2 4
```

```
prompt> gui_inspect_violations D2-4
```

SEE ALSO

`gui_wave_add_signal(2)`

`gui_violation_schematic_add_objects(2)`

gui_list_attrgroups

Lists attribute group information for a specified object type or all object types.

SYNTAX

```
status gui_list_attrgroups  
-class design_object  
-name name  
[-all]  
[-tcl]  
[-full]  
[-attr_list]
```

Data Types

```
design_object  string  
name          string
```

ARGUMENTS

-class *design_object*

Specifies the type of design object.

-name *name*

Specifies the name of the attribute group to be listed for the specified object type or for all object types.

-all

Lists the attribute groups for all object types. This option and the -class option are mutually exclusive.

-tcl

Lists the attribute group information in a tool command language (Tcl) format that is suitable for Tcl processing.

-full

Specifies the full Tcl output format, which reports all possible group information.

-attr_list

Lists attributes only.

DESCRIPTION

The **gui_list_attrgroups** command lists attribute group information for the specified object type or for all object types. Use the **-tcl** option to get results that are suitable for Tcl processing.

EXAMPLES

```
prompt> gui_list_attrgroups -class Cell -tcl  
"Hierarchy" "Edit"
```

```
prompt> gui_list_attrgroups -class Cell -name "Hierarchy" -tcl -attr_list  
{Name} {Hierarchical} {Owning Cell} {isPhysConstraint} {Hard Macro} ... {FullName}
```

```
prompt> gui_list_attrgroups -all -tcl  
Cell { "Hierarchy" "Edit" } Net { "Timing" } Pin { "Timing" }
```

```
prompt> gui_list_attrgroups -class Cell -tcl -attr_list  
{ "Hierarchy" { {Name} {Hierarchical} {Owning Cell} ...{FullName} } }  
{ "Edit" { {Name} {isPhysConstraint} {Orientation} ... {FullName} } }
```

```
prompt> gui_list_attrgroups -class Cell  
Cell, "Hierarchy", { {Name} {Hierarchical} {Owning Cell} {isPhysConstraint} ... } ;  
Cell, "Edit", { {Name} {isPhysConstraint} {Orientation} {Location} ... } ;  
The fields displayed above are in the following order:  
Class, Group Name, Attributes list;
```

SEE ALSO

```
gui_delete_attrgroup(2)  
gui_list_attrgroups(2)  
gui_update_attrgroup(2)
```

gui_list_category_rules

Lists all category rules except built-in rules by default.

SYNTAX

```
list gui_list_category_rules
    [-all | -names rule_names_list]
    [-format script | tcl_list]
```

Data Types

rule_names_list list

ARGUMENTS

-all

Lists all category rules, including built-in rules.

This option and the **-names** option are mutually exclusive. If you do not specify either of these options, the command lists all category rules except the built-in rules.

-names *rule_names_list*

Specifies the names of the category rules to be listed.

This option and the **-all** option are mutually exclusive. If you do not specify either of these options, the command lists all category rules except the built-in rules.

-format script | tcl_list

Specifies the output format in which the category rules are listed. The default output format is **script**.

When the **script** format is used, the category rules are listed as a Tool Command Language (Tcl) script of **gui_create_category_rule** commands that you can replay. In this case, the rules are streamed to the output stream. You can redirect the output by using the **redirect** command, for example, if you want to capture the script output in a file to use again later. When the **script** format is used, the command result is an empty string.

If the **tcl_list** format is specified, the category rules are listed in an "array set compatible" Tcl list format. In this case, the command result is a Tcl list.

DESCRIPTION

This command lists category rules that have already been created by the **gui_create_category_rule** command during the current run of the tool.

When you run this command without specifying either the **-all** option or the **-names** option, the command lists all category rules except built-in rules. Specify the **-all** option to list all category rules, including the built-in rules. Specify the **-names** option to list individual rules by name.

EXAMPLES

The following example lists all category rules, including the built-in rules:

```
prompt> gui_list_category_rules -all
gui_create_category_rule -name Pathgroups -builtin \
-category <path_group.full_name>
gui_create_category_rule -name Session -builtin \
-category <session_name>
gui_create_category_rule -name {Path Type} -builtin \
-category <path_type>
...
...
```

The following example lists all category rules except the built-in rules:

```
prompt> gui_create_category_rule -name rule1 -category <session_name>
rule1
prompt> gui_create_category_rule -name rule2 -category <path_group.full_name>
rule2
prompt> gui_list_category_rules
gui_create_category_rule -name rule1 \
-category <session_name>
gui_create_category_rule -name rule2 \
-category <path_group.full_name>
```

The following example redirects the script output to a file that you can source during a subsequent run of the tool:

```
prompt> redirect -file /remote/dir1/rules.tcl {gui_list_category_rules}
prompt>
```

The following example uses the **-names** option to list only the rules named rule1 and rule3:

```
prompt> gui_create_category_rule -name rule1 -category <session_name>
rule1
prompt> gui_create_category_rule -name rule2 -category <path_group.full_name>
rule2
prompt> gui_create_category_rule -name rule3 -category <startpoint.full_name>
rule3
prompt> gui_list_category_rules -names {rule1 rule3}
gui_create_category_rule -name rule1 \
-category <session_name>
gui_create_category_rule -name rule3 \
-category <startpoint.full_name>
```

The following example uses the **tcl_list** output format to query the category specification of the rule named rule1:

```
prompt> gui_create_category_rule -name rule1 -category <session_name>
```

```
rule1
prompt> gui_create_category_rule -name rule2 -category <path_group.full_name>
rule2
prompt> array set rules [gui_list_category_rules -format tcl_list]
Information: Defining new variable 'rules'. (CMD-041)
prompt> array set rule1_options $rules(rule1)
Information: Defining new variable 'rule1_options'. (CMD-041)
prompt> set rule1_cat_spec $rule1_options(category)
Information: Defining new variable 'rule1_cat_spec'. (CMD-041)
<session_name>
```

SEE ALSO

gui_create_category_rule(2)
gui_remove_category_rules(2)

gui_list_cell_block_marks

Lists the cell names and block mark values for cells that are marked in the design.

SYNTAX

```
list gui_list_cell_block_marks
```

ARGUMENTS

This command has no arguments

DESCRIPTION

This command lists the cell names and block mark values for all cells that are marked in the design. The listing is sorted by the full names of the marked cells.

EXAMPLES

The following example sets block marks on two hierarchical cell instances, and then lists the names of all the marked cells and their block marks:

```
prompt> gui_remove_cell_block_marks -all
prompt> gui_set_cell_block_marks I_TOP/I_ALU ALU
prompt> gui_set_cell_block_marks I_TOP/I_REG_FILE REG_FILE
prompt> gui_list_cell_block_marks
I_TOP/I_ALU  ALU
I_TOP/I_REG_FILE  REG_FILE
```

SEE ALSO

[gui_set_cell_block_marks\(2\)](#)
[gui_get_cell_block_marks\(2\)](#)

gui_remove_cell_block_marks(2)

gui_load_cell_density_mm

Loads the data for cell density map mode.

SYNTAX

```
status gui_load_cell_density_mm  
[-area area]
```

Data Types

area list

ARGUMENTS

-area *area*

Analyzes area for cell density. If area is not given, the default area is whole design area.

DESCRIPTION

The command generates the map mode view of cell density in a given area. Each grid is colored based on the percentage of occupied area of cells in the grid.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows cell density in the specified area.

```
prompt> gui_load_cell_density_mm -area \  
{284.820 390.670 1202.790 954.120}
```

SEE ALSO

gui_load_hierarchy_vm

Loads the data for hierarchy visual mode.

SYNTAX

```
status gui_load_hierarchy_vm  
-clear  
-level ncount  
-cells cell_list
```

Data Types

<i>level</i>	integer
<i>cell_list</i>	list

ARGUMENTS

-clear

Removes the coloring from all hierarchy levels.

-level *level*

Colors all the cells at a particular hierarchy level.

-cells *cell_list*

Colors specific hierarchy cells. If cells not provided, the default is cells in the entire design.

DESCRIPTION

Loads the data for hierarchy visual mode. The hierarchy visual mode provides a high-level view of the placement quality of logic blocks and hierarchical cells in your physical design. This visual mode can be used to color all the cells on a particular hierarchy level or just the specified hierarchical cells.

EXAMPLES

The following example clears the hierarchy visual mode.

```
prompt> gui_load_hierarchy_vm -clear
```

The following example colors all the cells at hierarchical level 2.

```
prompt> gui_load_hierarchy_vm -level 2
```

The following example colors all the cells in the selection.

```
prompt> gui_load_hierarchy_vm -cells [get_selection]
```

SEE ALSO

None.

gui_load_pin_density_mm

Loads the data for pin density map mode.

SYNTAX

```
status gui_load_pin_density_mm  
[-area area]
```

Data Types

area list

ARGUMENTS

-area *area*

Area to be analyzed for pin density. By default, the command uses the entire design area.

DESCRIPTION

The command generates map mode view of pin density in a given area. The grids are colored based on the number of pins within the grid.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows pin density in the specified area.

```
prompt> gui_load_pin_density_mm -area \  
{284.820 390.670 1202.790 954.120}
```

SEE ALSO

gui_load_voltage_area_vm

Loads the data for voltage area visual mode.

SYNTAX

status **gui_load_voltage_area_vm**

Data Types

ARGUMENTS

DESCRIPTION

The command generates the visual mode view of voltage areas. Each voltage area is colored based on the its type.

EXAMPLES

The following example will place all the voltage areas in the design into different bins based on their type.

```
prompt> gui_load_voltage_area_vm
```

SEE ALSO

None.

gui_mouse_tool

Operate mouse tools of a given view

SYNTAX

gui_mouse_tool

-window *window*
[-start *tool* |
-current |
-list |
-add_point {*x y*} |
-drag {{*x1 y1*} {*x2 y2*}} |
-delete_point |
-apply |
-reset |
-abort |
-cycle |
-cycle_back]

window *String*
tool *String*

ARGUMENTS

-window *window*

window specifies the window of which this command should operate the mouse tool.

-start *tool*

Starts the given mouse tool.

-current

Return the name of current active tool.

-list

Return the names of all available tools.

-add_point {*x y*}

Add an input point

-drag {{*x1 y1*} {*x2 y2*}}

Perform drag operation between two points.

-delete_point

Removes last input point

-apply

Performs tool action

-reset

Clears all pending input or ends the tool if no pending

-abort

Aborts mouse tool unconditionally

-cycle

Cycles through choices for a given operation

-cycle_back

Cycles back through choices for a given operation

DESCRIPTION

The **gui_mouse_tool** controls the mouse tool of a view that supports mouse actions log and replay capabilities. Not all views need to support all of the functionality provided by the interface of this command. User can use the command to script various mouse actions.

EXAMPLES

The following example has the layout view start the zoom in tool and perform a zoom to particular area.

```
> gui_mouse_tool -window Layout.1 -start ZOOM_IN_TOOL
> gui_mouse_tool -window Layout.1 -add_point {27.461 433.400}
> gui_mouse_tool -window Layout.1 -add_point {51.261 393.123}
> gui_mouse_tool -window Layout.1 -abort
```

SEE ALSO

gui_overlay_layout

Add/Remove/Adjust a design overlay to a layout window

SYNTAX

gui_overlay_layout

- window *windowID*
- design *design*
- add
- remove
- brightness *value*

windowID *String*

design *String*

add *Boolean flag*

remove *Boolean flag*

brightness *Integer*

ARGUMENTS

-window *windowID*

windowID specifies the layout window to add the overlay

-design *design*

design specifies the design to overlay

-add

add the design as an overlay

-remove

remove the design as an overlay

-brightness *value*

adjust the brightness of the overlay design

DESCRIPTION

Use this command to add, remove, or adjust an overlay on a layout view. The design to overlay must already be an opened design. You cannot overlay the same design twice nor can you overlay a design on itself. The -add and -remove flags are mutually exclusive. The brightness of an overlay can be varied from 0 (not visible at all) and 100 (fully bright). The -remove flag is mutually exclusive with the -brightness flags.

EXAMPLES

```
shell> gui_overlay_layout -window Layout.1 -design fill_view -add
shell> gui_overlay_layout -window Layout.1 -design fill_view -brightness Off
shell> gui_overlay_layout -window Layout.1 -design fill_view -brightness 33%
shell> gui_overlay_layout -window Layout.1 -design fill_view -remove
```

gui_query_objects

Get the value of a query text for a collection of object.

SYNTAX

```
string gui_query_objects  
      object_collection
```

Data Types

```
string object_collection
```

ARGUMENTS

object_collection

Specifies the collection of objects to use to get the query text.

DESCRIPTION

The **object_collection** command returns and displays the query text. Same text will be shown in Query toolbar for a particular object query.

A **object_collection** is collection of objects for which the query text will be generated. The collection of objects might be returned as the result of another command such as the **get_cells** command.

For information about collections, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example displays query text for the all plan groups in the design.

```
prompt> gui_query_objects [get_plan_groups]  
*****
```


Object summary:

plan_group : 5

plan_group : RES_I109

plan_group : RES_I110

plan_group : RES_I111

plan_group : RES_I112

plan_group : RES_I113

The following example displays a query text for cell named RES_I112/nr02d0_B3_1.

prompt> **gui_query_objects [get_cells RES_I112/nr02d0_B3_1]**

cell : RES_I112/nr02d0_B3_1

allowable_orientation	N FN FS S FW W E FE
area	25.600000
aspect_ratio	2.500000
bbox	{305.600 155.200} {308.800 163.200}
boundary	{305.600 155.200} {308.800 155.200} {308.800 163.200} {305.600 163.200} {305.600 155.200}
cell_id	3
design	test
eco_status	eco_reset
fp_area	25.600000
fp_aspect_ratio	2.500000
fp_height	8.000000
fp_number_of_hard_macro	0
fp_number_of_io_cell	0
fp_number_of_macro	0
fp_number_of_standard_cell	0
fp_width	3.200000
full_name	RES_I112/nr02d0_B3_1
height	8.000000
is_black_box	false
is_fixed	false
is_hard_macro	false
is_hierarchical	false
is_io	false
is_jtag	false
is_logical_black_box	false
is_macro_shape_fixed	false
is_mim	false
is_mim_master_instance	false
is_physical	true
is_physical_black_box	false
is_physical_only	false
is_placed	true
is_preserved	false
is_soft_fixed	false
is_soft_macro	false
is_spare_cell	false
mask_layout_type	std

movebound	RES_I112
name	nr02d0_B3_1
number_of_pins	3
object_class	cell
object_id	9782
orientation	N
origin	305.600 155.200
outer_keepout_margin	no_outer_keepout_margin
ref_lib_name	/remote/dtdata1/testdata/designs/syn/ggui/read_cell_demo/mw/cb25_typ
ref_name	nr02d0
ref_view_name	FRAM
sm_estimation_mode	unestimated
width	3.200000
within_ilm	false

SEE ALSO

[collections\(2\)](#)
[get_selection\(2\)](#)
[query_objects\(2\)](#)

gui_remove_all_annotations

Removes specified group of annotations or all annotations from the specified layout window or from all windows.

SYNTAX

```
status gui_remove_all_annotations  
[-window window_name]  
[-group group_name]
```

Data Types

```
window_name  string  
group_name  string
```

ARGUMENTS

-window *window_name*

Specifies the instance name of the layout window.

-group *group_name*

Specifies the annotation group name.

DESCRIPTION

This command removes specified group of annotations from the specified layout window. If group name is omitted, it will remove all annotations groups. If window name is omitted, it will remove annotations from all windows. The command returns 1 if it is successful.

EXAMPLES

```
prompt> gui_add_annotation -window Layout.1 -group Test1 \  
         -type rect -text Test1 -color green -width 2 \  
         -pattern Dense5Pattern {{200 200} {777 777}}  
1  
prompt> gui_remove_all_annotations -window Layout.1
```

1

SEE ALSO

`gui_add_annotation(2)`
`gui_remove_annotations(2)`

gui_remove_all_rulers

Removes rulers from the specified layout window or from all windows.

SYNTAX

```
status gui_remove_all_rulers  
[-window window_name]
```

Data Types

window_name string

ARGUMENTS

-window *window_name*

Specifies the instance name of the layout window.

DESCRIPTION

This command removes rulers from specified layout window, or from all windows if the **-window** option is omitted, and returns 1 if it is successful.

EXAMPLES

```
prompt> gui_remove_all_rulers -window Layout.1  
1
```

SEE ALSO

gui_remove_ruler(2)

gui_remove_annotations

Removes specified group of annotations from the specified layout window.

SYNTAX

```
status gui_remove_annotations  
[-window window_name]  
[-group group_name]  
[anno]
```

Data Types

```
window_name  string  
group_name  string  
anno        collection
```

ARGUMENTS

-window *window_name*

Specifies the instance name of the layout window.

-group *group_name*

Specifies the annotation group name.

anno

Specifies a collection of annotations to remove. Get a collection of annotations with the `gui_get_annotations` command. This option cannot be specified with any other option.

DESCRIPTION

This command removes specified group of annotations from the specified layout window. If group name is omitted, global group name is used. If window name is omitted, global window name is used. The command returns 1 if it is successful.

EXAMPLES

```
prompt> gui_add_annotation -window Layout.1 -group Test1 \  
-type rect -text Test1 -color green -width 2 \  
-pattern Dense5Pattern {{200 200} {777 777}}  
1  
prompt> gui_remove_annotations -window Layout.1  
1  
prompt> gui_remove_annotations [gui_get_annotations -filter client_data==MyData]  
1
```

SEE ALSO

gui_add_annotation(2)
gui_remove_all_annotations(2)
gui_get_annotations(2)

gui_remove_category_rules

Removes all category rules except built-in rules by default.

SYNTAX

```
string gui_remove_category_rules  
[-all | -names rule_names_list]
```

Data Types

<i>rule_names_list</i>	list

ARGUMENTS

-all

Removes all category rules, including built-in rules.

This option and the **-names** option are mutually exclusive. If you do not specify either of these options, the command removes all category rules except the built-in rules.

-names *rule_names_list*

Specifies the names of the category rules to be removed.

This option and the **-all** option are mutually exclusive. If you do not specify either of these options, the command removes all category rules except the built-in rules.

DESCRIPTION

This command removes category rules that have already been created by the **gui_create_category_rule** command during the current run of the tool.

When you use this command without specifying any options, it removes all category rules except built-in rules. Specify the **-all** option to remove all category rules, including the built-in rules. Specify the **-names** option to remove individual rules by name.

The command returns an empty string as its result.

EXAMPLES

The following example removes all category rules, including the built-in rules:

```
prompt> gui_remove_category_rules -all
```

The following example removes all category rules except the built-in rules:

```
prompt> gui_remove_category_rules
```

The following example removes the rules named Rule1 and Rule2:

```
prompt> gui_remove_category_rules -names {Rule1 Rule2}
```

The following example removes non-built-in rules at the top of a user-defined category rule creation script, which is being developed interactively. You can repeatedly change and refine this script by using a text editor, and you can source the script multiple times in a single run of the tool.

```
# first remove all existing user-defined non-built-in rules  
gui_remove_category_rules  
# now create the user-defined category rules  
gui_create_category_rule ...  
gui_create_category_rule ...  
and so forth
```

SEE ALSO

`gui_create_category_rule(2)`
`gui_list_category_rules(2)`

gui_remove_cell_block_marks

Removes the block mark string values from one or more cells.

SYNTAX

status **gui_remove_cell_block_marks**
-all | *cells*

Data Types

cells list

ARGUMENTS

-all

Removes all block marks on all cells that are marked.

This option and the *cells* option are mutually exclusive, and one of them must be specified.

cells

Removes block marks from one or more cells specified in a list of cell name patterns or cell collections.

This option and the **-all** option are mutually exclusive, and one of them must be specified.

DESCRIPTION

This command removes the block mark string values from one or more hierarchical or leaf-level cell instances. If the **-all** option is specified, then all the block marks are removed from all the cell instances that are marked. If the *cells* option is specified, then block marks are removed only from the specified cell instances.

EXAMPLES

The following example sets the block mark for a hierarchical cell instance identified by a cell name, and then removes the block mark from that cell instance:

```
prompt> gui_set_cell_block_marks I_TOP/I_ALU ALU
```

```
prompt> gui_remove_cell_block_marks I_TOP/I_ALU
```

The following example sets the block mark for a hierarchical cell instance identified by a nested run of the **get_cells** command, and then removes the block mark from that cell instance:

```
prompt> gui_set_cell_block_marks [get_cells I_TOP/I_ALU] ALU  
prompt> gui_remove_cell_block_marks [get_cells I_TOP/I_ALU]
```

The following example removes all the block marks from all the cell instances that are marked:

```
prompt> gui_remove_cell_block_marks -all
```

SEE ALSO

[gui_set_cell_block_marks\(2\)](#)
[gui_get_cell_block_marks\(2\)](#)
[gui_list_cell_block_marks\(2\)](#)

gui_remove_pref_key

Remove a preference key.

SYNTAX

```
string gui_remove_pref_key  
-key Key  
[-category Category]
```

ARGUMENTS

-key *Key*

Key specifies the key which will be used together with the specified category to uniquely identify the key to be removed.

-category *Category*

Category specifies the category that owns the specified key. If not specified, a default category will be used.

DESCRIPTION

This command removes the preference key specified by the key name.

SEE ALSO

```
gui_create_pref_category(2)  
gui_create_pref_key(2)  
gui_set_pref_value(2)  
gui_get_pref_value(2)  
gui_get_pref_keys(2)  
gui_exist_pref_category(2)  
gui_exist_pref_key(2)  
gui_update_pref_file(2)
```

gui_remove_ruler

Removes the specified rulers.

SYNTAX

```
status gui_remove_ruler  
[-window window_name]  
-point point | -all
```

Data Types

```
window_name  string  
point        point
```

ARGUMENTS

-window *window_name*

Specifies the instance name of the layout window.

-point *point*

Specifies the coordinates of the point near the ruler to be removed, in following format:

```
{x y}
```

-all

Removes all rulers.

DESCRIPTION

This command removes rulers from the specified layout window, or from all windows if the **-window** option is omitted, and returns 1 if it is successful.

EXAMPLES

```
prompt> gui_remove_ruler -window Layout.1 -point {400 400}
```

```
1  
prompt> gui_remove_ruler -window Layout.1 -all  
1
```

SEE ALSO

`gui_remove_all_rulers(2)`

gui_report_hotkeys

Report on the current hotkey bindings

SYNTAX

```
string gui_report_hotkeys  
[-window_type WindowTypeName]
```

WindowTypeName *String*

ARGUMENTS

-window_type *WindowTypeName*

WindowTypeName specifies the type of top level window that the hotkey should apply to. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable `gui_default_window_type(3)` specifies the default window type for the application.

DESCRIPTION

This command prints a report on the current key bindings. The list of bindings printed in the report are sorted by hotkey.

EXAMPLES

The following example creates a menu item and add an additional hotkey to it.

```
gui> gui_report_hotkeys
```

```
*****  
Report : hotkeys  
Window : LayoutWindow  
Version: V-2004.06-GALILEO-BETA1-1  
Date   : Tue Aug 31 10:37:17 2004  
*****
```

```
Hot Key   Type  Function
```

```
-----  
Ctrl+O  Menu  File->Open Design...  
Ctrl+S  Menu  File->Save Design  
Ctrl+R  Menu  Edit->Properties  
M       Menu  Edit->Move...  
C       Menu  Edit->Copy...  
Ctrl+F  Menu  View->Zoom->Zoom Full  
F       Menu  View->Zoom->Zoom Full  
P       Tcl  query_objects [get_selection]  
...
```

SEE ALSO

gui_set_hotkey(2)
gui_create_menu(2)
gui_delete_menu(2)
gui_create_toolbar(2)
gui_delete_toolbar(2)
gui_create_toolbar_item(2)
gui_delete_toolbar_item(2)
gui_default_window_type(3)
gui_custom_setup_files(3)

gui_report_task

task.

SYNTAX

```
string gui_report_task  
-task string  
| -item_root TaskName / ItemRootName  
[-file string]
```

ARGUMENTS

-task *string*

TaskName specifies the name of a task on which to report. This option is mutually exclusive with the -item_root option.

-item_root *TaskName | ItemRootName*

ItemRootName specifies the name of a task item root name on which to report. This option is mutually exclusive with the -task option.

-file *string*

FileName optionally specifies the output file into which the report is written.

DESCRIPTION

This command outputs a textual report of the given task to the xterm console. If -file is given, then the report is also output to the given file. The report will have the following approximate format. The strings bracketed with the angle brackets will be replaced with the actual attribute value.

Attributes are omitted if they are not defined. Item are shown one to a line with the hierarchy structure indicated by indentations and bars. The leaf-level task item names are followed by the associated task page name:

```
Task: <task name>  
Default: <true | false>  
Items: <items>
```

SEE ALSO

`gui_create_task(2)`
`gui_get_task_list(2)`

gui_schematic_add_logic

Adds logic to a schematic

SYNTAX

```
string gui_schematic_add_logic [-window <win>]  
    [-new]  
    [-schematic <schem>]  
    objs
```

```
string <win>  
string <schem>  
string objs
```

ARGUMENTS

-window <win>

Top level window name to place new schematic (default: current window). This option is ignored unless -new is specified.

-new

Create new schematic.

-schematic <schem>

Schematic view to update (default: most recently active schematic).

objs

Objects to add to the schematic.

DESCRIPTION

This command adds logic into a Path Schematic view. The command either creates a new view containing the specified objects or updates an existing view. The command always returns the name of the schematic that was updated or created.

EXAMPLES

The following example creates a new path schematic containing all the cells in the current design:

```
gui_schematic_add_logic -new [get_cells *]
```

SEE ALSO

```
gui_get_current_window(2)  
gui_schematic_remove_logic(2)
```

gui_schematic_remove_logic

Removes logic from a schematic

SYNTAX

```
string gui_schematic_remove_logic [-schematic <schem>]  
objs
```

```
string <schem>  
string objs
```

ARGUMENTS

-schematic <*schem*>

Schematic view to update (default: most recently active schematic).

objs

Objects to remove from the schematic.

DESCRIPTION

This command removes logic into a Path Schematic view. The command returns the name of the schematic that was updated

EXAMPLES

The following example removes the cell "U1" from the most recently active schematic view:

```
gui_schematic_remove_logic [get_cells U1]
```

SEE ALSO

```
gui_get_current_window(2)  
gui_schematic_add_logic(2)
```

gui_scroll

Scroll a window's viewport.

SYNTAX

gui_scroll

-window *window*
[-selection | -clct *clct* | [-habs *absX* | -hrel *relX*] [-vabs *absY* | -vrel *relY*]]

window *String*

absX *Float*

relX *Float*

absY *Float*

relY *Float*

ARGUMENTS

-window *window*

window specifies the window to be scrolled.

-selection

Determine the bounding box for all selected objects present in the view, and scroll such that the center of that box is visible.

-clct *clct*

Determine the bounding box for the collection objects present in the view, and scroll such that the center of that box is visible.

-habs *absX*

Scroll such that the specified X model position is centered in the viewport.

-hrel *relX*

Scroll horizontally by the specified number of pages. A page is the width of the viewport. Negative values scroll left and positive values scroll right.

-vabs *absY*

Scroll such that the specified Y model position is centered in the viewport.

-vrel *relY*

Scroll vertically by the specified number of pages. A page is the height of the viewport. Negative values scroll up and positive values scroll down.

DESCRIPTION

The **gui_scroll** command adjusts the position of views that support zooming. The viewport can be adjusted in the X and/or Y direction. Absolute values are expressed in model coordinates. Relative values allow paging. For example, *-hrel 1.0* means page to the right by the width of the viewport.

EXAMPLES

Scroll such that model coordinates (287.0, 1422.0) are centered in the viewport.

```
gui_scroll -window Layout.1 -habs 287.0 -vabs 1422.0
```

Scroll to the right by one viewport width.

```
gui_scroll -window Layout.1 -hrel 1.0
```

SEE ALSO

`gui_zoom(2)`

gui_select_vmbucket

Select the Contents of a Visual Mode Bucket

SYNTAX

```
string gui_select_vmbucket -vmname mode_identifier  
-name bucket_identifier  
[-replace | -add | -remove ]
```

```
string mode_identifier  
string bucket_identifier
```

ARGUMENTS

-vmname *mode_identifier*

Specifies the visual mode to which the bucket is a member. The visual mode must already exist. To see the current list of defined visual modes, use the **gui_list_vm** command.

-name *bucket_identifier*

Specifies the name of the visual mode bucket.

-replace

Optional argument that indicates that the contents of the visual mode bucket will replace the selection list.

-add

Optional string that indicates that the contents of the visual mode bucket will be added to the selection list.

-remove

Optional string that indicates the contents of the visual mode bucket will be removed from the selection list.

DESCRIPTION

The **gui_select_vmbucket** command selects the contents of a visual mode bucket. The visual mode bucket is a member of a specific visual mode. The contents of the visual mode bucket can replace, be added to, or be removed from the selection list.

EXAMPLES

Select the 2nd bucket of SNAPSHOT visual mode.

```
shell> gui_select_vmbucket -vmname SNAPSHOT -name 1 -replace
```

SEE ALSO

- gui_create_vm(2)
- gui_get_vm(2)
- gui_get_vmbucket(2)
- gui_remove_vm(2)
- gui_remove_vmbucket(2)
- gui_set_vm(2)
- gui_set_vmbucket(2)
- gui_update_vm(2)

gui_set_active_window

Make the specified window the active window

SYNTAX

status **gui_set_active_window**
-window *window_id*

Data Types

window_id string

ARGUMENTS

-window *window_id*

Specifies the window name id.

The matching window of this id will be made the active window.

Note: both toplevel windows and child windows of a particular toplevel window have a currently active window. If the window is a toplevel window then the currently active toplevel window is changed, of the window is a child window the currently active child window in the specified window's parent is changed.

DESCRIPTION

This command makes the specified window the active window.

EXAMPLES

The following example will make the toplevel window Toplevel.2 active.

```
shell> gui_set_active_window -window Toplevel.2
```

SEE ALSO

gui_close_window(2)
gui_exist_window(2)
gui_show_window(2)
gui_get_window_types(2)
gui_get_window_ids(2)
gui_create_window(2)
gui_get_current_window(2)

gui_set_bucket_option

Sets the value for an option on a bucket in the specified visual mode or map mode.

SYNTAX

```
string gui_set_bucket_option  
-map map_name  
-bucket bucket_name  
-option option_name  
-value value | -default
```

Data Types

```
map_name      string  
bucket_name   string  
option_name   string  
value         string
```

ARGUMENTS

-map *map_name*

Specifies the name of the visual mode or map mode.

-bucket *bucket_name*

Specifies the name of the bucket.

-option *option_name*

Specifies the name of the option to be set.

-value *value*

Specifies the value for the option.

The **-value** option is mutually exclusive with the **-default** option. You must use one, but not both.

-default

Sets the option to its default value.

The **-default** option is mutually exclusive with the **-value** option. You must use one, but not both.

DESCRIPTION

This command sets the value for an option on a bucket in a visual mode or map mode. You must specify the visual mode or map mode name, the bucket name, and the option name. You must set the option to either a specific value or its default value. Use the **-value** option to set a specific value or use the **-default** option to set the option to its default value.

EXAMPLES

The following example sets the color of *bucket1* to yellow in the global route congestion map:

```
prompt> gui_set_bucket_option -map AREAPARTITION \  
         -bucket bucket1 -option color -value yellow
```

SEE ALSO

```
gui_get_bucket_option(2)  
gui_get_bucket_option_list(2)  
gui_get_map_list(2)  
gui_get_map_option(2)  
gui_get_map_option_list(2)  
gui_set_map_option(2)
```

gui_set_cell_block_marks

Sets a block mark on one or more cells.

SYNTAX

```
status gui_set_cell_block_marks  
  cells  
  block_mark
```

Data Types

```
cells      list  
block_mark string
```

ARGUMENTS

cells

Lists one or more cells to mark with a block mark, in a list of cell name patterns or cell collections.

block_mark

Specifies the block mark string value to be set for the listed cells.

DESCRIPTION

This command sets a block mark for one or more hierarchical or leaf-level cell instances. Typically, a short symbolic string name is used as the block mark. Ultimately, a block mark can appear in the GUI as the text label for a timing path category generated from a dynamic category rule. Such block marks should be kept short to avoid using up too much screen space in the GUI.

A common application involves marking certain interesting intellectual property (IP) blocks within a design. These block marks affect the calculation of the following timing path shell attributes, which are defined only when the GUI is running:

- **blocks** - An ordered block level path; includes start, through, and end blocks
- **through_blocks** - Includes through blocks, but not the start and end blocks
- **start_block** - The start block
- **end_block** - The end block
- **start_end_blocks** - A pair consisting of the start block and the end block, in that order

- **start_end_blocks_sorted** - A pair consisting of the start block and the end block, sorted alphabetically

These block marks also affect the calculation of the following cell shell attribute, which is defined only when the GUI is running:

- **block_mark** - String value of a cell instance's (explicit) block mark, or an empty string if there is no explicit block mark

In calculating the timing path shell attributes listed previously, a block is calculated for each timing point of a timing path. If the leaf cell instance on which a timing point resides is explicitly marked with a block mark, by using the **gui_set_cell_block_marks** command, the block for the timing point is the block mark for the leaf cell. If the leaf cell instance on which a timing point resides is not explicitly marked, the block for the timing point is the block mark on the first explicitly marked hierarchical cell located by traversing up the logical hierarchy from that leaf cell instance.

If no direct or indirect hierarchical parent cell is found to be explicitly marked, the block for the timing point has a default implicit string value of "_Top_". For a timing point that is a startpoint or endpoint of a timing path, where the startpoint or endpoint is an external design port, the block has a default implicit string value of "_Port_".

Note: Each timing point of a timing path always has a defined block, which can be an explicit block mark, "_Top_", or "_Port_".

The **through_blocks** attribute value can be an empty set of blocks, which is represented by the string value "_Empty_".

EXAMPLES

The following example sets the block mark to ALU for a hierarchical cell instance identified by a cell name:

```
prompt> gui_set_cell_block_marks I_TOP/I_ALU ALU
```

The following example sets the block mark to ALU for a hierarchical cell instance identified by a nested run of the **get_cells** command:

```
prompt> gui_set_cell_block_marks [get_cells I_TOP/I_ALU] ALU
```

The following example sets the block mark for a list of cell instances where the first item is identified by a cell name and the second item is identified by a nested run of the **get_cells** command:

```
prompt> gui_set_cell_block_marks [list I_ALU [get_cells I_STACK_TOP/I1_STACK_MEM]] \
    my_mark
```

SEE ALSO

gui_get_cell_block_marks(2)
gui_remove_cell_block_marks(2)

gui_set_current_task

Set current task to the given task.

SYNTAX

```
string gui_set_current_task  
-task string
```

ARGUMENTS

DESCRIPTION

The `gui_set_current_task` command sets the current application task to the given task.

SEE ALSO

`gui_create_task(2)`
`gui_get_current_task(2)`
`gui_get_task_list(2)`

gui_set_highlight_options

Change the options that control highlighting.

SYNTAX

```
string gui_set_highlight_options  
  [-current_color color_id  
  | -next_color  
  | -auto_cycle_color enable]
```

Data Types

<i>color_id</i>	string
<i>enable</i>	bool

ARGUMENTS

-current_color *color_id*

Changes the current highlight color to the specified value. The current highlight color is used as a default when no color is specified for some operations.

-next_color

Changes the current highlight color to the next color in the set of available colors. This will wrap around so that it can be used to cycle through the color set. The provided colors are yellow, orange, red, green, blue, purple, light_orange, light_red, light_green, light_blue, and light_purple.

-auto_cycle_color *enable*

Specify if the current color should automatically be incremented after each gui_change_highlight -add operation. The color is not advanced if an explicit color is specified with the -color option of gui_change_highlight.

DESCRIPTION

The gui_set_highlight_options command can be used to modify highlighting behavior.

EXAMPLES

Set the current highlight color to blue.

```
prompt> gui_set_highlight_options -current_color blue
```

Enable auto cycling.

```
prompt> gui_set_highlight_options -auto_cycle_color true
```

Advance to the next color.

```
prompt> gui_set_highlight_options -next_color
```

SEE ALSO

`gui_change_highlight(2)`

`gui_get_highlight(2)`

`gui_get_highlight_options(2)`

gui_set_hotkey

Sets a key binding to a Tcl command or a menu command in a GUI window.

SYNTAX

```
string gui_set_hotkey
-hot_key key_name
-tcl_cmd command_name | -menu menu_name
[-replace]
[-delete]
[-replay_log_only]
[-window_type window_type_name | -all_window_types]
```

```
key_name          string
command_name      string
menu_name         string
window_type_name string
```

ARGUMENTS

-hot_key *key_name*

Specifies the key that you are associating with the specified command. The *key_name* is a string that contains the key name can also contain one or more modifiers. The valid modifier keys are **SHIFT**, **ALT**, and **CTRL**. You specify the modifier names by prepending them to the key name and connecting them with a plus sign (+).

If you bind an unmodified key and the shift-modified key is not bound, the binding works for both the shifted and nonshifted keys. If you set separate bindings for the shift-modified key and the unmodified key, the bindings are unique. Note that shift modifiers work only with letter keys and function keys.

Menus display the first key binding defined for a menu item. The key binding always appears in uppercase with its modifiers. For example, an unmodified accelerator for the letter "a" is shown as "A" and a shift-modified "a" (an uppercase A) is shown as "SHIFT+A."

-tcl_cmd *command_name*

Specifies the tool command language (Tcl) code that the tool executes when you press the key and its modifiers, if any.

The **-tcl_cmd** option and the **-menu** option are mutually exclusive.

-menu *menu_name*

Specifies the menu command that the tool invokes if you press the key and its modifiers, if any, when the menu command is enabled.

The *menu_name* is a string that specifies the hierarchy of the menu labels, with the labels connected by arrows formed from a

hyphen and a greater than sign (->). For example, "File->Exit" specifies the Exit command on the File menu. The *menu_name* can also include the mnemonic for the menu text, which is specified with an embedded ampersand (&), but they are not required to do so.

The **-menu** option and the **-tcl_cmd** option are mutually exclusive.

-replace

Replaces an existing key binding for the specified menu command or Tcl command with the specified key.

-delete

Removes the key binding from the specified menu command or Tcl command.

-replay_log_only

Limits logging of the Tcl command associated with the key to the command replay log file. The tool suppresses the command echo and result display to the xterm and console, and the command does not appear in the output log or the Tcl history log.

The tool ignores this option if you do not specify the **-tcl_cmd** option.

-window_type window_type_name

Specifies the type of top-level window that the key should apply to. (A top-level window is a window with a menu bar.) If you do not specify this option, the tool uses the application default window type.

The read-only variable **gui_default_window_type(3)** specifies the application default window type.

-all_window_types

Sets the key binding in every type of window for which it is applicable.

For menu command key bindings, the tool searches all of the top-level windows for the specified menu command and adds the key binding in any window where it exists. If the tool does not find any applicable windows, it issues a warning.

For Tcl command key bindings, the tool adds the binding in every type of top-level window.

DESCRIPTION

This command binds a given key to a function in the graphical user interface (GUI). The function might be available on a menu or specified by Tcl code. A menu command can have multiple key bindings. Key bindings can be specified for built-in or user-defined menu commands.

Key bindings for menu commands have several additional features that Tcl command key binding do not have. The function for a menu command key binding is invoked only when the menu command is enabled. This ensures that the function is invoked only when it is valid to invoke the function. In addition, the primary key bindings for menu commands appear with the menu text to make it easier for you to remember the bindings.

EXAMPLES

The following example creates a menu item and adds an additional key binding to it.

```
prompt> gui_create_menu -menu "&Test->Sub&Menu->Echo Hi" \
```

```
-tcl_cmd "echo hi" -hot_key "Ctrl+5"  
prompt> gui_set_hotkey -menu "Test->SubMenu->Echo Hi" \  
-hot_key "Ctrl+Y"
```

The following example creates a key binding, P, that invokes a Tcl command to print a query on the currently selected objects.

```
prompt> gui_set_hotkey -tcl_cmd {query_objects [get_selection]} \  
-key P
```

SEE ALSO

- gui_report_hotkeys(2)
- gui_create_menu(2)
- gui_delete_menu(2)
- gui_create_toolbar(2)
- gui_delete_toolbar(2)
- gui_create_toolbar_item(2)
- gui_delete_toolbar_item(2)
- gui_default_window_type(3)
- gui_custom_setup_files(3)

gui_set_layout_user_command

Set user defined command for layout input.

SYNTAX

```
status gui_set_layout_user_command  
-apply_cmd TclCmd | -clear  
[-input_type InputType]  
[-snap_type SnapType]  
[-status_text string]  
[-cancel_cmd TclCmd]
```

Data Types

TclCmd *string*
InputType one of [rectangle | line | polygon | point]
SnapType one of [litho | site | midsite | wiretrack | midwiretrack | user]

ARGUMENTS

-apply_cmd *TclCmd*

The user procedure to be called when input completed. The coordinates will be passed as an argument to this procedure. The options is mutually exclusive with -clear.

-clear

Cancel all pending input and return layout to default mouse mode. The options is mutually exclusive with -apply_cmd.

-input_type *InputType*

The desired input type: rectangle | line | polygon | point. Default is rectangle

-snap_type *SnapType*

The desired snap type: litho | site | midsite | wiretrack | midwiretrack | user Default is litho

-status_text *string*

The help string to be shown in layout window status bar.

-cancel_cmd *TclCmd*

The user procedure to be called when input is canceled.

DESCRIPTION

This command is used to facilitate creation of user defined tools to extend the layout view standard capabilities. The command sets layout view to given input mode and executes user callback procedure on input complete. The input points are passed as an argument to user callback.

EXAMPLES

Cut the object shape with user specified rectangle

```
proc my_cut_object_shape_proc { rect } {  
    change_selection [cut_objects [get_selection] -bbox {$rect}]  
}
```

```
gui_set_layout_user_command -apply_cmd {my_cut_object_shape_proc} -status_text "Drag the rectangle to cut selected objects"  
1
```

SEE ALSO

change_selection(2)
get_selection(2)

gui_set_map_option

Sets the value for an option in the specified visual mode or map mode.

SYNTAX

```
string gui_set_map_option  
-map map_name  
-option option_name  
-value value | -default
```

Data Types

```
map_name    string  
option_name string  
value       string
```

ARGUMENTS

-map *map_name*

Specifies the name of the visual mode or map mode.

-option *option_name*

Specifies the name of the option to be set.

-value *value*

Specifies the value for the option.

The **-value** option is mutually exclusive with the **-default** option. You must specify one, but not both.

-default

Sets the option to its default value.

The **-default** option is mutually exclusive with the **-value** option. You must specify one, but not both.

DESCRIPTION

This command sets the value for an option in a visual mode or map mode. You must specify the visual mode or map mode name and the option name. You must set the option to a specific value or its default value. Use the **-value** option to set a specific value or

the **-default** option to set the option to its default value.

EXAMPLES

The following example sets the number of bins to 9 in the global route congestion map:

```
prompt> gui_set_map_option -map AREAPARTITION \  
       -option num_bins -value 9
```

SEE ALSO

- `gui_get_bucket_option(2)`
- `gui_get_bucket_option_list(2)`
- `gui_get_map_list(2)`
- `gui_get_map_option(2)`
- `gui_get_map_option_list(2)`
- `gui_set_bucket_option(2)`

gui_set_mouse_tool_option

Set an option on a mouse tool

SYNTAX

```
string gui_set_mouse_tool_option -tool string  
-option string  
-value string
```

```
string string  
string string  
string string
```

ARGUMENTS

-tool *string*

Tool to set option for.

-option *string*

Option to set.

-value *string*

New option value.

DESCRIPTION

This command sets a mouse tool option. Mouse tool options influence the operation of the mouse tool when that tool is active

EXAMPLES

```
> gui_set_mouse_tool_option -tool SELECT -option InputMode -value Rectangle
```

SEE ALSO

`gui_mouse_tool(2)`
`gui_get_mouse_tool_option(2)`

gui_set_pref_value

Set the value of a preference key.

SYNTAX

```
string gui_set_pref_value  
-key Key -value Value  
[-category Category]
```

ARGUMENTS

-key *Key*

Key specifies the key to set the new value.

-value *Value*

Value specifies the new value of the key. The new value should be of the same data type, one of (integer,bool,double,color,string) as the data type of the value of the key.

-category *Category*

Category specifies the category that owns the specified key. If not specified, a default category will be used.

DESCRIPTION

This command set the value of a preference key. Returns the value of the preference key.

EXAMPLES

The following example create a user-defined category *some_category* for storing preferences and then create a boolean key *some_key* under that category with the initial value of false. The command **gui_set_pref_value** is then used to change the value of the preference key to true.

```
gui_create_pref_category -category some_category  
gui_create_pref_key -category my_category -key some_key -value_type bool -value false  
gui_set_pref_value -category my_category -key some_key -value true
```

SEE ALSO

gui_create_pref_category(2)
gui_create_pref_key(2)
gui_get_pref_value(2)
gui_remove_pref_key(2)
gui_get_pref_keys(2)
gui_exist_pref_category(2)
gui_exist_pref_key(2)
gui_update_pref_file(2)

gui_set_region

Sets the coordinates of the current region rectangle or rectilinear polygon.

SYNTAX

```
status gui_set_region  
      shape
```

Data Types

```
shape      list
```

ARGUMENTS

shape

Specifies the region coordinates.

For a rectangle, the values should be given in the following format:

```
{{x1 y1} {x2 y2}}
```

For a rectilinear polygon, the values should be given in the following format:

```
{{x1 y1} {x2 y2} ... {xN yN}}
```

DESCRIPTION

This command sets the region shape to be used in layout region mode and returns 1 if it is successful.

EXAMPLES

```
prompt> gui_set_region {{-1743.945 18.357} {42.833 1523.657}}  
1
```

SEE ALSO

`gui_get_region(2)`

gui_set_setting

Sets a setting on the specified window.

SYNTAX

gui_set_setting
-window *WindowID*
-setting *Setting*
-value *Value*

ARGUMENTS

-window *WindowID*

WindowID specifies the window to set the setting

-setting *Setting*

Setting specifies the setting to set

-value *Value*

Value specifies the value of the setting

DESCRIPTION

Sets a setting on the specified window. Windows are views or top level windows. The setting is a property of the window and the value's type is specific to the given setting.

EXAMPLES

```
prompt> gui_set_setting -window Layout.1 -setting showCell -value true
```

```
prompt> gui_set_setting -window Layout.1 -setting viewLevel -value 1
```

SEE ALSO

`gui_get_setting(2)`

gui_set_task_list

Set the list of tasks that are visible in the task assistant, and set their order.

SYNTAX

```
gui_set_task_list  
-tasks TaskList
```

DESCRIPTION

ARGUMENTS

-tasks *TaskList*

TaskList specifies the list of tasks that will be visible in the task assistant and their order. Tasks which are not included in the list will still exist but will not be shown in the selection combo box of the task assistant.

DESCRIPTION

The `gui_set_task_list` command sets the list of tasks that are visible and available in the task assistant and sets their order.

EXAMPLES

The following simple example sets the visible list of tasks to "Design Planning" followed by "Routing".

```
gui_set_task_list -tasks {{Design Planning} {Routing}}
```

SEE ALSO

`gui_get_task_list(2)`

gui_create_task(2)

gui_set_timing_table_paths

Set timing paths into a specified timing table.

SYNTAX

```
status gui_set_timing_table_paths  
[ -command ACmd ]  
[ -window AWindow ]  
[ -new ]  
[ -name ATable ]
```

Data Types

ACmd string
AWindow string
ATable string

ARGUMENTS

-command *ACmd*

Specifies a tcl command that returns a collection of timing paths.

If no command is specified, the "Select Paths" dialog will appear, where the desired timing paths could be specified.

-window *AWindow*

Specifies a top level window id where the timing table is open.

If this option is not specified the most recently used top level window will be used.

-new

Creates a new timing table.

This option cannot be used with option -name at the same time.

-name *ATable*

Specifies an existing timing table by its window id.

This option cannot be used with option -new at the same time.

DESCRIPTION

The **gui_set_timing_table_paths** command sets a collection of timing paths into a specified timing table. The command allows to create new timing tables in a specified top level window, or setting a table with a collection of timing paths which are generated by a specified command.

EXAMPLES

The following example opens a new timing table in the most recently used top level window with fifteen timing paths.

```
shell> gui_set_timing_table_paths -new -command {get_timing_paths -max_paths 15}
```

This example shows how to update an existing timing table called TimingAnalysisDriver.3 with a command to show the selected timing paths.

```
shell> set my_paths [get_timing_paths -nworst 20 -delay_type max_rise]
```

```
shell> change_selection $my_paths
```

```
shell> gui_set_timing_table_paths -name TimingAnalysisDriver.3 -command { get_selection }
```

SEE ALSO

[gui_get_window_ids\(2\)](#)

[get_timing_paths\(2\)](#)

gui_set_window_pref_key

Create a preference key owned by a particular window or window type

SYNTAX

```
new_value gui_set_window_pref_key
{-window window_id
 | -window_type window_type}
[-category category]
-key key
-value_type value_type
-value value
```

Data Types

```
window_id    string
window_type  string
category     string
key          string
value_type   string
value        string
```

ARGUMENTS

-window *window_id*

Specifies the name of the window that the preference will be applied to. This option is mutually exclusive with the -window_type option.

-window_type *window_type*

Specifies the name of the window type that instances of the type will have the preference be applied to. This option is mutually exclusive with the -window option.

-category *category*

Specifies the category that the key belongs to. If not specified, a default category will be assigned.

-key *key*

Specifies the name of the key to create.

-value_type *value_type*

Specifies the the data type of the value of the key. It must be one of [bool | integer | double | color | string].

-value *value*

Specifies the value of the key. The value should be set appropriately in accordance with its value type. The bool type accepts [true, false, 0, 1, on, off]. The color type accepts a named color, eg. "red" or a string specifying the color in this format "#RRGGBB", for example, "#0000FF".

RETURNS

new_value

The new value of the key.

DESCRIPTION

This command creates a preference key with the specified key name or sets the value of an existing pref key. This key will have the specified value type and value, and it will be created under the specified category. If the category is not specified, the key will belong to a default category. Returns the value of the preference key.

EXAMPLES

The following example will create a window and set a preference on that window.

```
\prompt> set wnd [gui_create_window -type Console]
\prompt> gui_set_window_pref_key -window $wnd -category {caption} -key {title}
-value_type string -value {Command Console}
```

SEE ALSO

- gui_get_window_pref_value(2)
- gui_get_window_pref_categories(2)
- gui_get_window_pref_keys(2)
- gui_create_pref_category(2)
- gui_set_pref_value(2)
- gui_get_pref_value(2)
- gui_remove_pref_key(2)
- gui_get_pref_keys(2)
- gui_exist_pref_category(2)
- gui_exist_pref_key(2)
- gui_update_pref_file(2)

gui_show_file_in_editor

Show the contents of a file in an external text editor.

SYNTAX

```
gui_show_file_in_editor  
-filename Filename
```

FileName String

ARGUMENTS

-filename *Filename*

Filename specifies the name of the file to be edited.

DESCRIPTION

The `gui_show_file_in_editor` command opens a text editor to edit the specified file. The editor is a separate task so doesn't block the current application.

By default the file is edited using the text editor from the `EDITOR` environment variable in a xterm. If the `EDITOR` environment variable is not set then the 'vi' text editor is used.

Both the editor and terminal (xterm) are searched for on the user's path. If either the editor or terminal can not be found the command will fail.

SEE ALSO

`gui_show_url_in_browser(2)`

gui_show_man_page

Show a man page in the man browser

SYNTAX

string **gui_show_man_page**
topic [-apropos]

Data Types

topic string

ARGUMENTS

topic

Man page topic to be shown.

DESCRIPTION

This command will launch or raise the man page browser, and display the topic specified in it. If no topic is specified, then the HOME page for man page indexes will be shown. If the specified topic does not exactly match a given man page, then additional searching will be done for commands. If the topic matches one or more commands, then an index page will be generated showing all of the commands which match the specified topic pattern. If the topic doesn't match any commands, then we will use the topic as the initial part of a command and add a * to it and do command matching. Any completions for the command will be shown on an index page that can then be used to select a man page for viewing. If the -apropos option is specified, then the topic will be searched via the apropos command and the topics returned will be shown with thier brief help strings, and provide links the the man pages referenced.

EXAMPLES

To show the man page for the find command:

`gui_show_man_page find` To show the man page for all commands containing the word cell:

`gui_show_man_page *cell*` To show the man page for the get_attribute command using command completion:

`gui_show_man_page get_attr`

SEE ALSO

man(2)
apropos(2)

gui_show_map

Displays or hides the specified visual mode or map mode.

SYNTAX

```
string gui_show_map  
-map map_name  
-show true | false  
[-window window]
```

Data Types

```
map_name  string  
window    string
```

ARGUMENTS

-map *map_name*

Specifies the name of the visual mode or map mode.

-show true | false

Displays the visual mode or map mode when set to **true**, or hides the visual mode or map mode when set to **false**.

-window *window*

Specifies the name of layout window in which to display or hide the visual mode or map mode.

DESCRIPTION

This command displays or hides the specified visual mode or map mode. If the **-window** option is not specified, the command uses the most recently active layout window.

EXAMPLES

The following example displays the highlight visual mode in the most recently active layout window:

```
prompt> gui_show_map -map Highlight \  
       -show true
```

SEE ALSO

`gui_get_map_list(2)`

gui_show_palette

Shows the palette in the specified window state.

SYNTAX

```
status gui_show_palette
{ -name palette_id | -type palette_type }
[ -parent parent_name ]
[ -show_state window_state ]
[ -dock_edge dock_edge ]
[ -size {width height} ]
```

Data Types

```
palette_id  string
palette_type string
parent_name string
window_state string
dock_edge  string
width      float
height     float
```

ARGUMENTS

-name *palette_id*

Specifies the palette name id.

Any matching palette of this name will be shown.

This option precludes the **-type** option

-type *palette_type*

Specifies the palette type id. Any matching palette of this type will be shown.

If the *-parent* option is specified only palettes in the specified parent toplevel window are shown. If the *-parent* option is not specified all palettes of this type in all toplevel windows are shown.

This option precludes the **-name** option

-parent *parent_name*

Specifies the parent toplevel window name.

-show_state *window_state*

Specifies the window state to show the palette.

Legal states are **maximized**, **minimized**, **normal**, or **docked**.

The default is **normal**.

-dock_edge *dock_edge*

Specifies the edge to dock the palette.

Legal dock edges are **left**, **top**, **right**, or **bottom**.

Note: this option is only valid if the state is set to **docked**.

The default is **right**.

-size {*width height*}

Specifies the width and height of the palette.

Note: this option is only valid if the state is set to **normal**.

DESCRIPTION

This command shows the specified palette in the specified state and optionally sets the geometry.

If no state is specified then it defaults to normal.

EXAMPLES

The following example shows the Layer palette docked on the left of the workspace.

```
shell> set top [gui_create_window -type TopLevel]
shell> set layout [gui_create_palette -type Layout -parent $top]
shell> gui_show_palette -name $layout -show_state docked -dock_edge left
```

SEE ALSO

`gui_hide_palette(2)`

gui_show_toolbar

Displays the specified toolbar or all the toolbars in the specified window.

SYNTAX

```
void gui_show_toolbar  
  -toolbar tool_bar_name | -all  
  [-window window_id]
```

Data Types

```
tool_bar_name  string  
window_id     string
```

ARGUMENTS

-toolbar *tool_bar_name*

Specifies the toolbar you want to display. The **-toolbar** option and the **-all** option are mutually exclusive.

-all

Displays all the toolbars in the specified window. The **-all** option and the **-toolbar** option are mutually exclusive.

-window *window_id*

Specifies the window in which you want to display the specified toolbar or all toolbars. By default, the tool displays the toolbar or toolbars in the active window.

DESCRIPTION

This command displays either all the toolbars or the toolbar with the specified name in the window with the specified window ID.

EXAMPLES

The following example displays the File toolbar in the active window:

```
prompt> gui_show_toolbar -toolbar File
```


The following example displays all the toolbars in the layout window named LayoutWindow.1:

```
prompt> gui_show_toolbar -all -window LayoutWindow.1
```

SEE ALSO

- `gui_create_toolbar(2)`
- `gui_delete_toolbar(2)`
- `gui_create_toolbar_item(2)`
- `gui_delete_toolbar_item(2)`
- `gui_hide_toolbar(2)`
- `gui_get_toolbar_names(2)`

gui_show_url_in_browser

Show the contents of a URL in an external web browser.

SYNTAX

gui_show_url_in_browser

-url *URL*

URL String

ARGUMENTS

-url *Url*

Url specifies the url to be displayed.

DESCRIPTION

The `gui_show_url_in_browser` command an external web browser to display the specified web page. The default web browser is defined by the application but is usually 'firefox' on UNIX platforms.

If the web browser supports external communication (which 'firefox' does) then if no web browser is running the web browser is started. If the web browser is running then the URL is loaded in the browser usually as a new tab.

The default browser can be set using the 'gui_online_browser' variable but the set of valid browsers is usually restricted by the application so not all browsers may be supported. If a unsupported browser is specified the command will fail.

SEE ALSO

`gui_online_browser(3)`

`gui_show_file_in_editor(2)`

gui_show_window

Show the window in the specified window state

SYNTAX

```
status gui_show_window  
-window window_id  
[ -show_state window_state ]  
[ { -rect rect | -size isize } ]
```

Data Types

```
window_id    string  
window_state string  
rect         {{x1 y1} {x2 y2}}  
isize        {width height}
```

ARGUMENTS

-window *window_id*

Specifies the toplevel window or view id to show.

-show_state *show_window_state*

Specifies the window state to show the toplevel window or view.

For a toplevel window we have one of:

- normal** - show at preferred size
- maximized** - show maximized (fill whole screen)
- minimized** - show iconized
- hidden** - hide but do not destroy

Note: The window manager may or may not fully honor the specified state. See your window manager documentation for more details.

For a view window we have one of:

- normal** - show at preferred size in view area and raise to top
- maximized** - show maximized in view area and raise to top
- minimized** - show iconized in view area
- hidden** - hide but do not destroy

Note: If displaying maximized then any existing view windows are also maximized. If displaying minimized or normal then any maximized windows are shown as normal.

The default is **SHOW_WINDOW_STATE_NORMAL**.

-rect *rect*

Specifies the size and position of the window.

It is of this format "{x1 y1} {x2 y2}", where {x1 y1} specifies the top left location and {x2 y2} specifies the bottom right location of the window.

Note: This option precludes the use of the *-size* option.

-size *isize*

Specifies the width and height of the window.

It is of this format "{width height}"

Note: This option precludes the use of the *-rect* option.

DESCRIPTION

This command shows the specified toplevel window or view in the specified state and optionally sets the geometry.

If no state is specified then it defaults to normal.

EXAMPLES

The following examples will create a layout window then use this command to illustrate the various options

```
shell> set top [gui_create_window -type TopLevel]
shell> set layout [gui_create_window -type Layout -parent $top]
shell> gui_show_window -window $layout -show_state {maximized}
shell> gui_show_window -window $layout -show_state {normal}
```

SEE ALSO

gui_close_window(2)
gui_exist_window(2)
gui_create_window(2)
gui_get_window_types(2)
gui_get_window_ids(2)
gui_set_active_window(2)
gui_get_current_window(2)

gui_start

Starts the application GUI.

SYNTAX

```
string gui_start  
[-file script]  
[-no_windows]
```

Data Types

script string

ARGUMENTS

-file *script*

The given script file is sourced before the GUI starts.

-no_windows

The GUI starts without showing the default window.

DESCRIPTION

This command starts the application GUI from the shell prompt. It is ignored if the application GUI has already been started. `start_gui` is an alias for `gui_start`.

Note the application can only ever connect to one X server during program execution, so changing the value of the `DISPLAY` environment variable after the first successful `gui_start` command has no effect.

EXAMPLES

The following example starts the application GUI.

```
prompt> gui_start
```

SEE ALSO

gui_stop(2)
start_gui(2)
stop_gui(2)

gui_stop

Stops the application GUI.

SYNTAX

string **gui_stop**

ARGUMENTS

None.

DESCRIPTION

This command stops the application GUI and returns to the shell prompt. It is ignored if the application GUI has not been started or has been stopped. stop_gui is an alias for gui_stop.

EXAMPLES

The following example stops the application GUI and returns to the shell prompt.

```
prompt> gui_stop
```

SEE ALSO

gui_start(2)
start_gui(2)
stop_gui(2)

gui_update_attrgroup

Updates a group of attributes for an object type.

SYNTAX

```
status gui_update_attrgroup
-class design_object
-name name
[-attr_list {{attr_1}...{attr_n}}]
[-add]
[-delete]
[-move up | down | top | bottom | after | before]
[-attr attribute]
[-anchor attribute]
```

Data Types

```
design_object  string
name          string
attr_1        string
attribute     string
```

ARGUMENTS

-class *design_object*

Specifies the type of design object.

-name *name*

Specifies the non-editable name of the attribute group to update for the specified object type.

-attr_list {{*attr_1*}...{*attr_n*}}

Lists and orders the attributes to be included in the group.

-add

Adds the attribute specified by the **-attr** option to the end of attributes list, by default, or after the attribute specified by the **-anchor** option.

-delete

Removes the attributes specified by the **-attr** option from the attribute list. Note that the attribute still exists.

-move up | down | top | bottom | after | before

Moves the attribute specified by the **-attr** option in the specified direction. If you specify **before** or **after** to move the attribute relative to the position of a reference attribute, use the **-anchor** option to specify the reference attribute.

-attr *attribute*

Specifies the attribute to be added, removed, or moved with the **-add**, **-delete**, or **-move** option.

-anchor *attribute*

Specifies the reference attribute to be used with the **-add** or **-move** option.

DESCRIPTION

The **gui_update_attrgroup** command updates an attribute group for the specified object type. You can reset the entire attribute list by using the **-attr_list** option, or you can change the list by adding, removing, or moving with a single attribute.

The order of the attributes in an attribute group is important and is set by the **-attr_list** option when you create or update the group.

EXAMPLES

```
prompt> gui_update_attrgroup -class Cell -name "TestGroup2" \  
-attr_list { "TestAttr1" "TestAttr2" "TestAttr3" }
```

```
prompt> gui_update_attrgroup -class Cell -name "TestGroup2" \  
-add -attr "TestAttr5"
```

```
prompt> gui_update_attrgroup -class Cell -name "TestGroup2" \  
-add -attr "TestAttr5" -anchor "TestAttr2"
```

```
prompt> gui_update_attrgroup -class Cell -name "TestGroup2" \  
-delete -attr "TestAttr5"
```

```
prompt> gui_update_attrgroup -class Cell -name "TestGroup2" \  
-move down -attr "TestAttr2"
```

SEE ALSO

gui_create_attrgroup(2)
gui_delete_attrgroup(2)
gui_list_attrgroups(2)

gui_update_pref_file

Save current application preferences to the user preference file.

SYNTAX

```
string gui_update_pref_file  
[-file FilePathName]
```

ARGUMENTS

-file *FilePathName*

FilePathName specifies the full pathname of the user preference file to write the current preferences to. If this option is missing, the file to write to is controlled by the gui variables : pref_file_name and pref_file_path. These two variables are set with the gui_set_var command and their values retrieved with the gui_get_var command.

DESCRIPTION

This command updates the user preference file. Normally, application will automatically save user preferences to a default system preference file on exit, and retrieve the preferences from the default user preference file on application start. So, this command is rarely used, except internally.

SEE ALSO

```
gui_create_pref_category(2)  
gui_create_pref_key(2)  
gui_set_pref_value(2)  
gui_get_pref_value(2)  
gui_remove_pref_key(2)  
gui_get_pref_keys(2)  
gui_exist_pref_category(2)  
gui_exist_pref_key(2)
```

gui_view_port_history

Does the gui_view_port_history operations.

SYNTAX

```
gui_view_port_history
  [-window fwindow_name]
  [-next]
  [-previous]
  [-add name]
  [-to name]
  [-list_names]
  [-rect bounding box]
  [-delete_name]
  [-isprevious]
  [-isnext]
  [-dialog]
  [-tcl_list]
  [-help_cmd]
```

ARGUMENTS

-window *fwindow_name*

Give the name of a window. this is required option except if you use *-dialog* option.

-next

Specify if you want to go to previous stored view port history. If you use *-next* option you can not use any other option except *-window* option: they are mutually exclusive.

-previous

Specify if you want to go to previous stored view port history. If you use *-previous* option you can not use any other option except *-window* option: they are mutually exclusive.

-add *name*

Current view port will be stored by this name. *-add* option can be combined with *-rect* option. In that case the view port history with the given rect instead of current view port will be saved in to view port history with the given name.

If you use *-add* option you can not use any other option except *-window* and *-rect* option : they are mutually exclusive.

-to *name*

Give the name of a view port history already stored. If you use *-to* option you can not use any other option except *-window* option: they are mutually exclusive.

-list_names

Specify if you want to see list of names of all the named view port history. If you use *-list_names* option you can not use any other option except *-window* option: they are mutually exclusive.

-delete_name *name*

The already stored view port history with this name will be deleted. If you use *-delete_name* option you can not use any other option except *-window* option: they are mutually exclusive.

-rect *boundingbox*

Adds give the bounding box (in design units) to unnamed view port history. The values should be given in following format. format: {{x1 y1} {x2 y2}} *-rect* option can be combined with *-add* option. In that case the view port history with the given rect instead of current view port will be saved in to view port history with the given name. If you use *-add* option you can not use any other option except *-window* and *-add* option : they are mutually exclusive.

-isnext

return true returns true if there previous zoom history. If you use *-isnext* option you can not use any other option except *-window* option: they are mutually exclusive.

-isprevious

returns true if there next zoom history. If you use *-isprevious* option you can not use any other option except *-window* option: they are mutually exclusive.

-dialog

specify to invoke dialog for named view port history. If you use *-dialog* option you can not use any other option except *-help_cmd* option: they are mutually exclusive.

-tcl_list

return a tcl_list of named view port history If you use *-tcl_list* option you can not use any other option except *-window* option: they are mutually exclusive.

help_cmd *help_command*

Give a command which will be executed when help button is clicked in the dialog.

DESCRIPTION

This command store view port for the application. At least one option needs to be specified.

last 100 View ports will be stored in view port history, which can be accessed by specifying **-previous** and **-next** options.

When you use **-add** then the current view port will be stored by the given name in the to view port history.

when you use **-to** then the view port stored by this name will be shown and added to the view port history.

EXAMPLES

The following example zooms in to the given rectangle.

```
fpc_shell> gui_view_port_history -rect {{-2232.321 -536.887} {141.286 2175.807}}
```

The following example zooms to next .

```
fpc_shell> gui_view_port_history -next
```

The following example zooms to previous.

```
fpc_shell> gui_view_port_history -previous
```

The following example gives list of all the named zooms.

```
fpc_shell> gui_view_port_history -list_names
```

The following example puts the current zoom in to zoom history by given name.

```
fpc_shell> gui_view_port_history -add xyz
```

The following example zooms to already stored to zoom .

```
fpc_shell> gui_view_port_history -to xyz
```

The following example deletes an already stored to zoom.

```
fpc_shell> gui_view_port_history -delete xyz
```

gui_violation_schematic_add_objects

Adds the specified objects to the violation schematic in a violation inspector window.

SYNTAX

```
status gui_violation_schematic_add_objects  
[-window window_name]  
[-clct]  
object_list
```

Data Types

window_name string
object_list list

ARGUMENTS

-window *window_name*

Specifies the violation inspector window in which you want to add the specified objects. If *window_name* does not exist, the command exits with an error message.

By default, the command adds the objects to the most recently active violation inspector window.

-clct

Causes the command to treat *object_list* as a collection of object handles. By default, the command treats *object_list* as a list of object names.

object_list

Specifies the objects that you want to add to the schematic in the violation inspector window. By default, *object_list* is a list of object names. If you use the **-clct** option, *object_list* specifies a collection of object handles. In both cases, *object_list* can consist of heterogeneous names.

DESCRIPTION

This command adds the specified objects (leaf cells, pins, or nets) to the violation schematic in the specified or active violation inspector view window.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example adds a cell named gate_clock to the violation schematic in the active violation inspector window:

```
prompt> gui_violation_schematic_add_objects gate_clock
```

The following example adds a cell named gate_clk and a port named i_rd to the violation schematic in the violation inspector window named violationInspector.2:

```
prompt> gui_violation_schematic_add_objects \  
-window violationInspector.2 {gate_clk i_rd}
```

The following example adds the selected objects to the violation inspector window named ViolationInspector.3:

```
prompt> gui_violation_schematic_add_objects \  
-window ViolationInspector.3 -clct [get_selection]
```

SEE ALSO

gui_inspect_violations(2)

gui_wave_add_signal

Adds the specified signals to the waveform view in a violation inspector window.

SYNTAX

```
status gui_wave_add_signal
[-window window_name]
[-clct]
object_list
```

Data Types

```
window_name    string
object_list    list
```

ARGUMENTS

-window *window_name*

Specifies the violation inspector window with the waveform view in which you want to add the specified signals. If *window_name* does not exist, the command exits with an error message.

By default, the command adds the signals to the first open violation inspector window with a waveform view.

-clct

Causes the command to treat *object_list* as a collection of object handles. By default, the command treats *object_list* as a list of object names.

object_list

Specifies the signals that you want to add to the waveform view in the violation inspector window. You can specify leaf cells, pins, ports, nets, or buses.

By default, *object_list* is a list of object names. If you use the **-clct** option, *object_list* specifies a collection of object handles. In both cases, *object_list* can consist of heterogeneous names.

DESCRIPTION

This command adds the specified signals to the waveform view in a violation inspector window. The command creates a new group that contains a list of the specified signals. If you specify a cell, the group contains all the pins on the cell. If you specify a bus, the group contains all the nets in the bus.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example adds the port named i_rd to the waveform view in the first open violation inspector window:

```
prompt> gui_wave_add_signal i_rd
```

The following example adds the selected objects to the waveform view in the window named ViolationInspector.3:

```
prompt> gui_wave_add_signal -window ViolationInspector.3 \  
-clct [get_selection]
```

SEE ALSO

`gui_inspect_violations(2)`

gui_write_layout_image

Save a layout image in a file

SYNTAX

```
status gui_write_layout_image  
-output file_name  
[-window window_title]  
[-view_only]
```

Data Types

file_name string
window_title string

ARGUMENTS

-output *file_name*

Specifies the output file name, including a suffix (.jpg, .png, .xpm, or .bmp) that identifies format.

-window *window_title*

Identifies the layout window. The defaults is the most recently used Layout window.

-view_only

Writes only the layout view contents. The default is to write all of the layout window contents.

DESCRIPTION

The command `gui_write_layout_image` command writes the layout image, as seen on the screen, into a file. The image content can be the complete layout window (including layout window title, menus, menu bars and other visible toolbars docked in the layout window) or simply that of the view. The output file name must include one of supported formats as file suffix.

The supported formats are JPEG, PNG, XPM, and BMP.

This command works only in a GUI session.

EXAMPLES

The following example writes the latest layout image, excluding the layout window title, menus, and toolbars, in JPEG format to a file named `risc_core.jpg` that resides in the current directory.

```
prompt> gui_write_layout_image -view_only -output risc_core.jpg
```

SEE ALSO

`gui_start(2)`
`gui_stop(2)`

gui_write_window_image

Saves an image of a view window or top-level window in the specified image file.

SYNTAX

```
status gui_write_window_image  
-file file_name  
[ -format png | xpm | jpg | bmp ]  
[ -window window_id ]
```

Data Types

<i>file_name</i>	string
<i>window_id</i>	string

ARGUMENTS

-file *file_name*

Specifies the name of the file in which the tool saves the window image.

If you include a file name extension, it determines the image format unless you also specify the **-format** option.

-format png | xpm | jpg | bmp

Specifies the image format to be used in the file. The default value is png.

Note that if the format that you specify with the **-format** option is different from the format specified by the file name extension, the **-format** value takes precedence and the tool appends an additional extension to the file name.

-window *window_id*

Specifies the name of the window for which you want to save an image in the specified image file.

You can view a list of the window names for all the open top-level and view windows by using the **gui_get_window_ids** command.

By default, the tool saves an image of the most recently active view window.

DESCRIPTION

The **gui_write_window_image** command saves an image of a view window or a top-level window in the specified image file.

EXAMPLES

In the following example, the tool creates a new schematic view window and saves an XPM image of the window in a file named my_image.xpm:

```
prompt> set schm [gui_create_schematic]  
prompt> gui_write_window_image -file my_image.xpm -window $schm
```

SEE ALSO

- gui_create_schematic(2)
- gui_create_window(2)
- gui_get_window_ids(2)
- gui_set_active_window(2)
- gui_get_current_window(2)

gui_zoom

Change the viewport of a view

SYNTAX

gui_zoom

-window *window*
[-fit | -exact]
[-full]
[-selection]
[-rect *{lx ly} {ux uy}*]
[-factor *factor*]
[-at_point *{x y}*]
[-clct *c/c*]
[-zoom_in_mode]
[-zoom_out_mode]
[-select_mode]

window *String*
rect *String*
factor *Float*

ARGUMENTS

-window *window*

window specifies the window of which this command should change the viewport.

-full

Zoom such that all objects are visible.

-factor *factor*

Keep the center of the viewport and zoom by factor *factor*. A *factor* > 1 causes a zoom-in, a *factor* < 1 causes a zoom-out. The argument must be greater than zero.

-at_point *{x y}*

Can only be specified with -factor argument. If supplied, zoom is performed at specified point instead of viewport center. The option is not supported by all window types.

-fit

Effects the -rect, -selection and -clct arguments. If supplied, will cause the zoom to not over zoom. Will zoom a reasonable

distance away from the rectangle or objects being zoomed. The definition of "reasonable" is window dependent. This argument is mutually exclusive with `-exact`.

-exact

Effects the `-rect`, `-selection` and `-clct` arguments. If supplied, will cause the zoom to zoom exactly to the arguments. This argument is mutually exclusive with `-fit`.

-selection

Zoom such that all selected objects that are represented in the window are visible and if possible are easy to see and are shown in a reasonable context. If neither `-fit` or `-exact` is supplied, `-fit` is assumed.

-rect *{lx ly} {ux uy}*

Zoom such that the window shows the rectangle *rect* in its viewport. What coordinates are assumed for *rect* are dependent on the window. If neither `-fit` or `-exact` is supplied, `-exact` is assumed.

-clct *clct*

Zoom such that all objects in *clct* that are represented in the window are visible and if possible, are easy to see and are shown in a reasonable context. If neither `-fit` or `-exact` is supplied, `-fit` is assumed.

-zoom_in_mode

Put the window in zoom-in mode.

-zoom_out_mode

Put the window in zoom-out mode.

-select_mode

Put the window in selection mode.

DESCRIPTION

The **gui_zoom** controls the viewport of a view that supports zooming. Not all views need to support all of the functionality provided by the interface of this command.

EXAMPLES

The following example has the layout view show the entire design and then zooms in by a factor of 2, i.e. only a quarter of the area of the design is still visible:

```
gui_zoom -window Layout.1 -full
gui_zoom -window Layout.1 -factor 2
gui_zoom -window Layout.1 -rect {{0.0 0.0} {123.456 500.123}}
gui_zoom -window Layout.1 -rect {{0.0 0.0} {123.456 500.123}} -fit
gui_zoom -window Layout.1 -selection -exact
```

SEE ALSO

gui_zoom_all_layouts_to_current_view

Rescales all layout views to the current view.

SYNTAX

```
gui_zoom_all_layouts_to_current_view
```

ARGUMENTS

None.

DESCRIPTION

The command sets the zoom factor of all layout views to the zoom factor of current layout view.

EXAMPLES

```
prompt> gui_zoom_all_layouts_to_current_view
```

SEE ALSO

gui_zoom(2)

help

Displays quick help for one or more commands.

SYNTAX

```
string help  
  [-verbose]  
  [-groups]  
  [pattern]
```

Data Types

pattern string

ARGUMENTS

-verbose

Displays the command options; for example *command_name* **-help**.

-groups

Displays a list of command groups only.

pattern

Displays commands matching the specified pattern.

DESCRIPTION

The **help** command is used to get quick help for one or more commands or procedures. This is not the same as the **man** command that displays reference manual pages for a command. There are many levels of help.

By typing the **help** command, a brief informational message is printed followed by the available command groups.

List all of the commands in a group by typing the group name as the argument to **help**. Each command is followed by a one-line description of the command.

To get a one-line help description for a single command, type **help** followed by the command name. You can specify a wildcard pattern for the name; for example, all commands containing the string "alias". Use the **-verbose** option to get syntax help for one or more commands. Use the **-groups** option to show only the command groups in the application. This option cannot be combined with any other option.

EXAMPLES

The following example lists the commands by command group:

```
prompt> help
```

Specify a command name or wild card pattern to get help on individual commands. Use the '-verbose' option to get detailed option help for a command. Commands also provide help when the '-help' option is passed to the command.'

You can also specify a command group to get the commands available in that group. Available groups are:

```
Procedures:      Miscellaneous procedures
Help:             Help commands
Builtins:         Generic Tcl commands
...
```

The following example displays the list of procedures in the Procedures group:

```
prompt> help procedures
```

```
ls          # List files
sh          # Execute a shell command
```

The following example uses a wildcard character to display a one-line description of all commands beginning with *a*:

```
prompt> help a*
```

```
alias       # Create a command which expands to words.
append      # Builtin
array       # Builtin
```

This example displays option information for the **source** command:

```
prompt> help -verbose source
```

```
source      # Read a file and execute it as a script
[-echo]     (Echo all commands)
[-verbose]  (Display intermediate results)
file_name   (Script file to read)
```

SEE ALSO

man(2)
sh_help_shows_group_overview(3)

history

Displays or modifies the commands recorded in the history list.

SYNTAX

```
string history  
    [-h]  
    [-r]  
    [argument_list]
```

Data Types

argument_list list

ARGUMENTS

-h

Displays the history list without the leading numbers. You can use this for creating scripts from existing history. You can then source the script with the **source** command. You can combine this option with only a single numeric argument. Note that this option is a nonstandard extension to Tcl.

-r

Reverses the order of output so that most recent history entries display first rather than the oldest entries first. You can combine this option with only a single numeric argument. Note that this option is a nonstandard extension to Tcl.

argument_list

Additional arguments to **history** (see DESCRIPTION).

DESCRIPTION

The **history** command performs one of several operations related to recently-executed commands recorded in a history list. Each of these recorded commands is referred to as an "event." The most commonly used forms of the command are described below. You can combine each with either the **-h** or **-r** option, but not both.

- With no arguments, the **history** command returns a formatted string (intended for you to read) giving the event number and contents for each of the events in the history list.
- If a single, integer argument *count* is specified, only the most recent *count* events are returned. Note that this option is a nonstandard extension to Tcl.

- Initially, 20 events are retained in the history list. You can change the length of the history list using the following:

history keep *count*

Tcl supports many additional forms of the **history** command. See the "Advanced Tcl History" section below.

EXAMPLES

The following examples show the basic forms of the **history** command. The first is an example of how to limit the number of events shown using a single numeric argument:

```
prompt> history 3
7 set base_name "my_file"
8 set fname [format "%s.db" $base_name]
9 history 3
```

Using the **-r** option creates the history listing in reverse order:

```
prompt> history -r 3
9 history -r 3
8 set fname [format "%s.db" $base_name]
7 set base_name "my_file"
```

Using the **-h** option removes the leading numbers from each history line:

```
prompt> history -h 3
set base_name "my_file"
set fname [format "%s.db" $base_name]
history -h 3
```

Advanced Tcl History

The **history** command performs one of several operations related to recently-executed commands recorded in a history list. Each of these recorded commands is referred to as an "event." When specifying an event to the **history** command, the following forms may be used:

- A number, which if positive, refers to the event with that number (all events are numbered starting at 1). If the number is negative, it selects an event relative to the current event; for example, **-1** refers to the previous event, **-2** to the one before that, and so on. Event **0** refers to the current event.
- A string selects the most recent event that matches the string. An event is considered to match the string either if the string is the same as the first characters of the event, or if the string matches the event in the sense of the **string match** command.

The **history** command can take any of the following forms:

history

Same as **history info**, described below.

history add *command* [*exec*]

Adds the *command* argument to the history list as a new event. If **exec** is specified (or abbreviated), the command is also executed and its result is returned. If **exec** is not specified, an empty string is returned.

history change *newValue* [*event_number*]

Replaces the value recorded for an event with *newValue*. The *event_number* specifies the event to replace, and defaults to the

current event (not event -1). This command is intended for use in commands that implement new forms of history substitution and want to replace the current event (that invokes the substitution) with the command created through substitution. The return value is an empty string.

history clear

Erases the history list. The current keep limit is retained. The history event numbers are reset.

history event [*event_number*]

Returns the value of the event given by *event_number*. The default value of *event_number* is -1.

history info [*count*]

Returns a formatted string, giving the event number and contents for each of the events in the history list except the current event. If *count* is specified, then only the most recent *count* events are returned.

history keep [*count*]

Changes the size of the history list to *count* events. Initially, 20 events are retained in the history list. If *count* is not specified, the current keep limit is returned.

history nextid

Returns the number of the next event to be recorded in the history list. Use this for printing the event number in command-line prompts.

history redo [*event_number*]

Reruns the command indicated by *event* and returns its result. The default value of *event_number* is -1. This command results in history revision. See the following section for details.

History Revision

Pre-8.0 Tcl had a complex history revision mechanism. The current mechanism is more limited, and the **substitute** and **words** history operations have been removed. The **clear** operation was added.

The **redo** history option results in much simpler "history revision." When this option is invoked, the most recent event is modified to eliminate the history command and replace it with the result of the history command. If you want to redo an event without modifying the history, use the **event** operation to retrieve an event, and use the **add** operation to add it to history and execute it.

identify_clock_gating

Identifies Power Compiler-inserted clock-gating circuitry in a structural netlist.

SYNTAX

```
status identify_clock_gating  
[-gating_elements cell_collection]
```

Data Types

cell_collection collection

ARGUMENTS

-gating_elements *cell_collection*

Marks the specified cell as a gating element. By default, this option is off. Performs specific recognition of the listed cells as gating elements. Each element must meet the following minimum conditions to be accepted as a gating element:

- Minimum 2 input pins
 - Minimum 1 output pin
 - One of the inputs pins must be connected to a clock source
-

OBSOLETE ARGUMENTS

The following arguments are no longer supported and are ignored if used:

-reset_only *cells_or_pins*

-gated_element *gated_cells_or_pins*

-ungated_element *ungated_cells*

DESCRIPTION

This command identifies the Power Compiler-inserted clock-gating circuitry in the structural netlist. Identification refers to the

process of detecting clock gates and the corresponding gated element association and setting different attributes on these objects. These attributes enable the later steps in the tool to work efficiently.

There is no need to invoke this command if the netlist was in .ddc or Milkyway format and the clock-gating structure was not changed outside the tool.

Set the correct clock-gating style by using the **set_clock_gating_style** command before invoking the **identify_clock_gating** command.

The most common usage is to invoke the command without any options. The **identify_clock_gating** command searches all of the clocks in the design for clock gates and gated cells. The tool then marks the clock-gating properties with attributes that are used by subsequent clock gating commands. Because only clocks are traversed in this case, it is important to run the **create_clock** command before running the **identify_clock_gating** command in this mode.

If test control pins are present on the clock gates, the **identify_clock_gating** command tries to reinstate the necessary attributes for the **insert_dft** command to work. The control signal is assumed to be of **scan_enable** type, unless otherwise marked as **test_mode**, by using the **set_clock_gating_style** command with the **-control_signal test_mode** option, before running the **identify_clock_gating** command. These attributes are applied to all the clock gates of the design that have been identified.

If you specify the **-gating_elements** option, the command automatically detects the gated elements through netlist traversal. Each successfully identified element is automatically constrained with the **pwr_cg_preservation_type** attribute to prevent it from being removed from the design and from being merged with other gating elements. If the specified gating cell has the characteristics of a type of clock gate that can be inserted by the tool, then the value for **pwr_cg_preservation_type** is **preserve**. If the gating cell is different from one originally inserted by Power Compiler, then the value for the attribute is **unmodifiable_read_only**. After the manual identification process is done, the command also performs the design-scoped identification process to search for additional clock gates, if any.

The **identify_clock_gating** command assumes the only nonsequential cells in the clock network are single output cells (netlinks, inverters, or buffers). If it finds a combinational cell with multiple outputs, it stops the traversal, with the exception of pad cells.

The **identify_clock_gating** command reports identified elements by printing two (2) informational messages: PWR-875 and PWR-877. The former reports the number of manually identified elements (only if **-gating_element** option is used) for cells that do not have the characteristics of a type of clock gate that can be inserted by the tool. The latter reports the number of elements identified by the design-scoped identification process.

In addition, you can use the **report_clock_gating** command to verify the results accuracy of the identification step. If there are any problems, rerun the **identify_clock_gating** command with the **-gating_elements** option to make repairs.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the typical flow for using the **identify_clock_gating** command to identify the clock-gating circuitry:

```
prompt> read_verilog mapped_design.v
prompt> current_design top
prompt> link
prompt> create_clock clk

prompt> identify_clock_gating
prompt> report_clock_gating
```

The following example shows how to apply clock-gating attributes to the user-inserted *user_cg1* clock gate as the corresponding

clock gate of the *reg1_reg* register bank:

```
prompt> read_verilog mapped_design.v
prompt> current_design top
prompt> link

prompt> identify_clock_gating -gating_elements user_cg1

prompt> report_clock_gating
```

The following example shows how to apply clock-gating attributes to the user-inserted *user_cg1* clock gate and allow the tool identify the corresponding registers:

```
prompt> read_verilog mapped_design.v
prompt> current_design top
prompt> link

prompt> identify_clock_gating -gating_elements user_cg1

prompt> report_clock_gating
```

The following example shows how to use the **identify_clock_gating** command with the **report_clock_gating** command to fix problems:

```
prompt> read_verilog mapped_design.v
prompt> current_design top
prompt> link

prompt> create_clock -period 3 clk
prompt> identify_clock_gating

# Inspect if all clock gates are identified
prompt> report_clock_gating -gated

# Assume cg1 is not identified as a clock gate
prompt> identify_clock_gating -gating_elements cg1

prompt> report_clock_gating -gated
```

SEE ALSO

```
create_clock(2)
insert_dft(2)
remove_clock_gating(2)
report_clock_gating(2)
rewire_clock_gating(2)
set_clock_gating_style(2)
set_preserve_clock_gate(2)
placer_gated_register_area_multiplier(3)
```

identify_register_banks

Identifies register banking opportunities and writes out a register banking script. This command is supported only in Design Compiler topographical mode.

SYNTAX

```
status identify_register_banks
  -output_file file_name
  [-input_map_file file_name]
  [-register_group_file file_name]
  [-maximum_flop_count flop_count]
  [-minimum_flop_count flop_count]
  [-exclude_instances exclude_cells]
  [-wns_threshold percentage]
  [-common_net_pins names]
  [-name_prefix prefix]
  [-wns_threshold_file file_name]
  [-exclude_library_cells library_cells]
  [-exclude_size_only_flops]
  [-exclude_start_stop_scan_flops]
  [-multibit_components_only]
```

Data Types

<i>file_name</i>	string
<i>flop_count</i>	integer
<i>exclude_cells</i>	list or collection
<i>percentage</i>	float
<i>names</i>	list of string
<i>prefix</i>	string
<i>library_cells</i>	list or collection

ARGUMENTS

-output_file *file_name*

Specifies the name of the output file that contains the generated register banking guidance commands. The output file contains a series of **create_register_bank** commands. The output file can be sourced back into the tool to replace single-bit registers with multibit registers.

-input_map_file *file_name*

Specifies the name of the input map file that contains the mapping information about which multibit register bank replaces which single-bit registers. The tool reads in the input map file during placement.

The format for the input map file is described in the FILE FORMATS section of this man page.

If the **-input_map_file** option is not specified, the **-register_group_file** option is ignored. If you do not specify the input map file and register group file, the **identify_register_banks** command identifies the single-bit registers that can be replaced by available multibit registers in the target and link libraries based on the functional information of the library cells; the tool then uses that information to control mapping.

-register_group_file *file_name*

Specifies the file from which functional grouping information for the library cells are to be read. If this option is not provided, the tool treats all library cells as part of one functional group.

The format for the register group file is described in the FILE FORMATS section of this man page.

If the **-register_group_file** option is not specified, the **-input_map_file** option is ignored. If you do not specify the input map file and register group file, the **identify_register_banks** command identifies the single-bit registers that can be replaced by available multibit registers in the target and link libraries based on the functional information of the library cells; the tool then uses that information to control mapping.

-maximum_flop_count *flop_count*

Specifies the maximum number of registers that can be grouped together. If this option is not provided, the tool automatically determines the value as the maximum bit-width for which mapping is provided in the input map file.

-minimum_flop_count *flop_count*

Specifies the minimum number of registers that can be grouped together. If this option is not provided, the tool automatically determines the value as the minimum bit-width for which mapping is provided in the input map file.

-exclude_instances *exclude_cells*

Specifies the list of instances to be ignored for grouping.

-wns_threshold *percentage*

Specifies the percentage of the total number of registers in the design that can be ignored during grouping if registers have negative setup slack.

For example, if there are 1000 registers in the design and the **-wns_threshold** option is specified as 10, the tool ignores 100 (10% of 1000) registers that have the most negative setup slack. If the design has fewer registers with negative setup slack, for example 60, the tool ignores all 60 registers.

The default is zero (no register is ignored). This option is ignored by the tool if the **-wns_threshold_file** option is specified.

-common_net_pins *names*

Specifies a list of pin names that need to be connected to the same net for all the registers in a group.

For example, if you specify the **-common_net_pins {CP RN}** option setting, the tool ensures that the CP pins of the registers in a group are connected to a common net, and that the RN pins of the registers are connected to a common net. In other words, the tool does not form a group with registers with CP pins that are connected to different nets or RN pins that are connected to different nets.

-name_prefix *prefix*

Specifies a prefix for the register bank name that the **create_register_bank** command uses in the output file. By default, the tool derives the name of the register bank by concatenating the names of the registers it is grouping.

For example, if you do not specify the **-name_prefix** option, the tool groups a_reg[0] and a_reg[1] registers to one multibit cell and generates the following in the output file:

```
create_register_bank -name a_reg[0]_a_reg[1]
```

If you specify the **-name_prefix** option with an MBIT prefix, the output file generated during grouping uses MBIT as the prefix, as shown:

```
create_register_bank -name MBIT_a_reg[0]_a_reg[1]
```

To revert to the previous naming style, before version J-2014.09-SP3, use the following variable setting:

```
set_app_var banking_enable_concatenate_name false
```

-wns_threshold_file *file_name*

Specifies the file from which the worst negative slack (WNS) threshold information (based on the timing group) is read. This overrides the **-wns_threshold** option.

If this option is not specified, the **-wns_threshold** option is used to determine the timing-critical registers that are ignored.

The format for the WNS threshold file is described in the FILE FORMATS section of this man page.

-exclude_library_cells *library_cells*

Specifies the list of library cells whose instances are ignored for grouping.

-exclude_size_only_flops

Specifies that flip-flops having size_only attribute should be excluded from grouping.

-exclude_start_stop_scan_flops

Specifies that flip-flops containing start or stop pins of scan chains should be excluded from grouping.

-multibit_components_only

Optional switch to make the clustering process work only on pre-existing multibit components of the design. This allows the user to avoid banking cells that don't belong to multibit components, as well as to avoid banking cells of different multibit components together in the same multibit cell.

DESCRIPTION

The **identify_register_banks** command sets the strategy used to identify groups of registers that can be replaced by multibit registers and generates a register banking script file. This script can then be sourced back into the tool to replace single-bit registers with multibit registers.

FILE FORMATS

Format for specifying -input_map_file

You can specify the input map file sequence as shown:

```
bits { number_of_instances ref_multibit_flop } { ... }
```

Where **bits** specifies the number of single-bit registers in a group to be replaced, *number_of_instance* specifies the number of multibit registers to be used, and *ref_multibit_flop* specifies the reference multibit library cell to be used.

For example:

```
4 {1 MREG4}
10 {2 MREG4} {1 MREG2}
```

4 {1 MREG4} specifies that a group of four single-bit registers can be replaced by one cell whose reference is MREG4.

Similarly, **10 {2 MREG4} {1 MREG2}** means a group of ten single-bit registers can be replaced using two instances of MREG4 and one instance of MREG2.

You can include a comment using the # character in the input map file. Lines beginning with the # character to the end of the line is treated as a comment.

Format for specifying -register_group_file

You can specify the register group file sequence as shown:

```
reg_group_name number {list_of_reference_of_single_bit_flops} {list_of_reference_of_multi_bit_flops}
```

Where *reg_group_name* specifies the name of the register group, *number* specifies the number of reference of single-bit registers used for this register group, *list_of_reference_of_single_bit_flops* specifies the list of references of single-bit registers, and *list_of_reference_of_multi_bit_flops* specifies the list of references of multibit registers.

For example:

```
reg_group_1 3 {REGX1 REGX2 REGX4} {MREG2 MREG4}
reg_group_2 3 {REGNX1 REGNX2 REGNX4} {MREG2N MREG4N}
```

reg_group_1 3 {REGX1 REGX2 REGX4} {MREG2 MREG4} means that single-bit registers whose reference is either REGX1, REGX2, or REGX4 can be grouped together and replaced with a multibit bank whose reference is either MREG2 or MREG4.

Format for specifying -wns_threshold_file

You can specify the WNS threshold file sequence as shown:

```
name : threshold_number
```

Where *name* is either the name of a timing path group or the keyword **others**, and *threshold_number* is a floating-point number.

For example:

```
to_memory : 23.3
rsg_ack : 15
others : 10.0
```

In this example, 23.3% of the registers having the most negative setup slack in the timing path group **to_memory** are ignored for grouping. Similarly, 15% of registers in the timing path group **rsg_ack** are ignored for grouping, and 10% of the registers for the rest of the timing path groups in the design are ignored for grouping.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **identify_register_banks** command to generate the register banking script:

```
prompt> identify_register_banks -input_map_file ./input.map -output_file ./create_bank.out
1
```

SEE ALSO

`create_register_bank(2)`
`split_register_bank(2)`

if

Conditional execution control structure.

SYNTAX

```
return_val if
```

ARGUMENTS

The **if** command has no arguments.

DESCRIPTION

The command conditionally executes commands based on the value of an evaluated expression. The format of the **if** command is similar to that in the C programming language.

The expression is evaluated upon execution. If the expression evaluates to a nonzero value, the **then** part of the command is executed. If the expression evaluates to zero, the optional **else** part of the command is executed. If **else** is not present, no commands execute.

The **if** command allows an **else-if** form. In this form, the condition expressions are evaluated in sequence, until one having a nonzero value is encountered. The *statement_block* corresponding to the nonzero condition executes; if none of the conditions evaluate to a nonzero value, the *else_statement_block* executes as expected.

The return value of the **if** command is the return value of the last statement executed. If no statements execute, the return value is 0.

The entire output of the **if** command can be redirected to a file. This file contains the output of any of the statements executed in the body of the **if** command. See the Tcl documentation for more information on file redirection.

If you use **else** in the **if** command, **else** must appear on the same line as the closing brace. If **else** is not on the same line as the closing brace, it is flagged as a syntax error.

EXAMPLES

The following example shows a simple use of the **if** command.

```
if {$x == 0} {
```

```
    echo "Equal"  
  } elseif {$x > 0} {  
    echo "Greater"  
  } else {  
    echo "Less"  
  }
```

SEE ALSO

break(2)
continue(2)
while(2)

index_collection

Given a collection and an index into it, if the index is in range, create a new collection containing only the single object at the index in the base collection. The base collection remains unchanged.

Optionally, a second index can be passed which creates a new collection with the objects between the two indices in the base collection. Makes an implicit assumption that $\text{index} \leq \text{index2}$

SYNTAX

```
collection index_collection
  collection1
  index
  index2
```

Data Types

<i>collection1</i>	collection
<i>index</i>	index2
<i>index2</i>	index

ARGUMENTS

collection1

Specifies the collection to be searched.

index

Specifies the index into the collection.

The index is either a simple integer to specify the offset from the start of the collection or the string "end" optionally followed by a negative integer to specify the offset from the end of the collection.

For the start offset case the value must range from 0 to **sizeof_collection** - 1. For the end offset case the negative integer (if specified) must range from 0 to -(**sizeof_collection** - 1).

index2

Specifies an optional second index into the collection.

Specifies a second index to create a collection of a range of objects from the base collection, defined by the two indices [*index*, *index2*] (boundaries inclusive)

DESCRIPTION

You can use the **index_collection** command to extract a single object from a collection. The result is a new collection containing only that object. The index operation is done in constant time - it is independent of the number of elements in the collection, or the specific index.

The range of indices is from 0 to one less than the size of the collection. If the specified index is outside that range, an error message is generated.

Commands that create a collection of objects do not impose a specific order on the collection, but they do generate the objects in the same, predictable order each time. Applications that support the sorting of collections allow you to impose a specific order on a collection.

You can use the empty string for the *collection1* argument. However, by definition, any index into the empty collection is invalid. Therefore, using the **index_collection** command with the empty collection always generates the empty collection as a result and generates an error message.

Note that not all collections can be indexed.

EXAMPLES

The following examples from PrimeTime use the **index_collection** command to extract the first, second, last and penultimate object of a collection.

```
prompt> set c1 [get_cells {u1 u2 u3}]
{u1 u2 u3}
prompt> query_objects [index_collection $c1 0]
{u1}
prompt> query_objects [index_collection $c1 1]
{u2}
prompt> query_objects [index_collection $c1 0 1]
{u1 u2}
prompt> query_objects [index_collection $c1 end]
{u3}
prompt> query_objects [index_collection $c1 end-1]
{u2}
prompt> query_objects [index_collection $c1 end-1 end]
{u2 u3}
```

SEE ALSO

collections(2)
query_objects(2)
sizeof_collection(2)

infer_switching_activity

Infers and sets the switching activity annotation on drivers of special pins of the current design.

SYNTAX

```
int infer_switching_activity
  [-apply]
  [-output file_name]
  [-nosplit]
  [-verbose]
  [-scenarios scenario_list]
  [-sci_based type]
```

Data Types

file_name string
scenario_list list

ARGUMENTS

-apply

Applies the proposed switching activity annotation on the drivers of special pins.

-output *file_name*

Writes the script to the specified file.

-nosplit

Prevents lines from being split when column fields overflow. Most of the information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line in the correct column.

-verbose

Displays the detailed special pin information.

-scenarios *scenario_list*

Specifies the scenarios to be considered. If this option is not specified, then only the current scenario is considered. Output of the command is generated separately for each scenario.

-sci_based *type*

Specifies whether to report or apply activity on any essential point based on the number of special control inputs it influences. The **type** specified can take one of the following three values: **sci**, **non_sci**, and **all**. If type is **sci**, the activity for any essential

point is reported and applied based on the number of SCIs (Special Control Inputs) in its fanout. If there is no SCI in the fanout of an essential point, the tool does not report for the point. For the **non_sci** type, the default activity value is reported and applied on all essential points that do not have any SCIs in their fanout. If the type is **all**, the tool reports and applies all the essential points. If this option is not specified, the default behavior corresponds to **all**.

DESCRIPTION

This command is used to infer the switching activity for the drivers of special pins. These special pins include asynchronous set, asynchronous clear, synchronous set, and synchronous clear. The switching activity include simple toggle rate and static probability on the drivers of these pins.

For statistics on the switching activity annotation on the current design, use the **report_saif** or **report_activity** command.

Multicorner-Multimode Support

This command uses information from the current scenario only if **-scenarios** option is not specified.

EXAMPLES

The following **infer_switching_activity** command, issued after **compile_ultra**, displays the current and proposed switching activity annotation for the driver of a special pin.

```
prompt> infer_switching_activity
infer_switching_activity
Created by infer_switching_activity on Sun Mar 20 12:32:16 2011
```

```
-----
              Current  Current  Proposed  Proposed
              Static   Toggle   Static   Toggle
Objects      Type      Probability Rate   Probability Rate
-----
rst          driver    None      None     1.0     0.0
-----
1
```

The following **infer_switching_activity** command, issued after **compile_ultra**, displays the current and proposed switching activity annotation for the driver of a special pin. It also displays the special pin information.

```
prompt> infer_switching_activity -verbose
infer_switching_activity
Created by infer_switching_activity on Fri Jul 15 14:40:28 2011

-----
      Current  Current Proposed  Proposed
      Static   Toggle Static   Toggle      Receiver
Driver Probability Rate Probability Rate Receiver  Type
-----
rst   None     None  1.0     0.0  cnt_reg[4]/CLRZ async_clear
                                cnt_reg[3]/CLRZ async_clear
                                cnt_reg[2]/CLRZ async_clear
```

```
cnt_reg[1]/CLRZ async_clear  
cnt_reg[0]/CLRZ async_clear
```

There are 5 receivers for rst.

1

SEE ALSO

get_switching_activity(2)
set_switching_activity(2)
reset_switching_activity(2)
report_activity(2)
report_power(2)

insert_buffer

Inserts buffer cells on specified nets or nets connected to specified ports or pins.

SYNTAX

```
collection insert_buffer  
  [-new_net_names new_net_names]  
  [-new_cell_names new_cell_names]  
  [-no_of_cells number]  
  [-inverter_pair]  
  object_list  
  buffer_lib_cell
```

Data Types

```
new_net_names    list  
new_cell_names  list  
number           integer  
object_list     list  
buffer_lib_cell collection
```

ARGUMENTS

-new_net_names *new_net_names*

Specifies the names of the new nets created by buffer insertion. You should specify one net name per buffer, or two net names per inverter pair when inserting inverter pairs. You can optionally specify only a single common base name; the tool generates new net names by adding unique numeric suffixes to this base name.

The specified names can be any valid net names, but must be leaf (not hierarchical) names and cannot contain embedded hierarchical separators. They must be unique in the current context, as specified by the current instance. If the specified net name already exists, the command adds a suffix of "_%d" or "_%d_%d" to the net name.

By default, the command uses the base name `eco_net`.

-new_cell_names *new_cell_names*

Specifies the names of the new buffer cells inserted. You should specify one cell name per buffer, or two cell names per inverter pair when inserting inverter pairs. You can optionally specify only a single common base name; the tool generates new cell names by adding unique numeric suffixes to this base name.

The specified names can be any valid cell names, but must be leaf (not hierarchical) names and cannot contain embedded hierarchical separators. They must be unique in the current context, as specified by the current instance. If the specified cell name already exists, the command adds a suffix of "_%d" or "_%d_%d" to the cell name.

By default, the command uses the common base name `eco_cell`.

-no_of_cells *number*

Specifies the number of buffer cells or inverter pairs to be inserted per net. The inserted repeaters are connected back-to-back in series. By default, the command inserts a single buffer cell or inverter pair per net.

-inverter_pair

Inserts inverter pairs instead of buffer cells. If you use this option, you need to also specify a library cell that has an inverting output, using the *buffer_lib_cell* option.

object_list

Specifies a list of nets, pins, or ports to be buffered. The new buffers or inverter pairs are placed in the specified net or in the net connected to the specified pins or ports, close to pin or port.

If you specify a net, the inserted buffer is the new load of the specified existing net.

If you specify pins, the tool groups all of the specified pins based on the nets to which they are connected. When the grouped pins are load pins, the tool inserts the buffers so that the new buffer cells can drive them. When the grouped pins are driver pins, the tool connects the new buffer cells so that they become the load of the specified driver pin.

buffer_lib_cell

Specifies the library cell object to be used as a buffer or inverter. Specify the object as either a named library cell or a library cell collection. If the library cell is a buffer cell, the number of instances inserted is equal to the number specified by the **-no_of_cells** option. If the library cell is an inverter, the cells are inserted as pairs, so the number of instances inserted is twice the number specified by the **-no_of_cells** option.

If the library cell has both inverting and noninverting outputs (that is, it can act as both a buffer and inverter), the **-inverter_pair** option controls which output is used. If the library cell has multiple outputs of the same type, the command uses the first noninverting or inverting output found.

DESCRIPTION

This command adds buffers or inverter pairs at one or more specified nets, pins, or ports. A library cell with a single input and one or more outputs can be used as the buffer or inverter, as long as each output has the same or inverted logic function of the input.

Like all other netlist editing commands, all of the **insert_buffer** command's arguments must be valid to have a successful command run. If any specified argument is invalid, the netlist remains unchanged. If the command succeeds, the result is a collection of the newly inserted cells. If the command fails, the result is an empty collection or an empty string.

By default, each newly created cell has a name beginning with the `eco_cell` string and ending with a unique numeric suffix. Each newly created net has a name beginning with the `eco_net` string and ending with a unique numeric suffix. To override the default name strings, use the **-new_net_names** and **-new_cell_names** options.

You can mimic buffer insertion by using other commands such as **create_cell**, **create_net**, **disconnect_net**, and **connect_net**. The **insert_buffer** command provides a more efficient and reliable way to insert buffers.

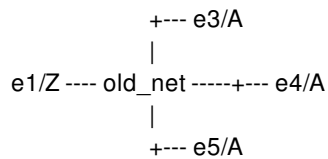
The **insert_buffer** command uses the following basic rules to check its arguments for validity:

- The pin or port must be in the current scope (at or below the current instance).
For a description of special scoping rules, see the "Buffering Inside Boundary Pins" section.
- The pin or port must be connected to a net.

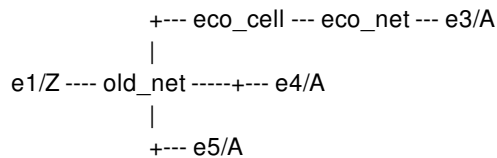
- Bidirectional pins cannot be buffered.
- The net cannot be a PG net or a tie net.
- The specified library cell cannot be a sequential device.
- The specified library cell must be a buffer or inverter, as previously defined.
- The library cell must have an inverting output when you use the **-inverter_pair** option.
The command uses the first inverting output and inserts two cells so that it preserves the logic sense of the path.
- The *number* argument of the **-no_of_cells** option must be a positive, nonzero integer.

Inserting and Connecting the New Buffer

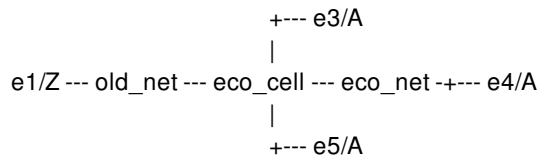
The following figure shows a network example that is used to describe the rules used by the **insert_buffer** command to connect the new buffer or inverter pair.



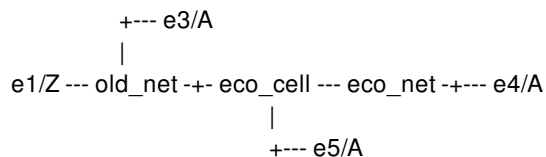
If the specified pin is a load, the load first is disconnected from its old net and then connected to the new net, as shown in the following figure:



If the specified pin is a driver, all loads on that net are disconnected from the old net and are connected to the new net, as shown in the following figure:



If you specify a list of load pins on the same net, these load pins are grouped and the new net drives them, as shown in the following figure:



The command cannot buffer a multidriver net.

Buffering Inside Boundary Pins

When you specify the insertion of a buffer at a pin on the boundary of a hierarchical block, the **insert_buffer** command inserts the buffer either inside or outside the hierarchical block, depending on the current hierarchical scope. To insert the buffer inside the hierarchical block, set the scope to that block by using the **current_instance** command, and then run the **insert_buffer**

command. The tool inserts the buffer within the block to which the **current_instance** is set.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies a library cell that is not a buffer. The command fails and generates an error message.

```
prompt> insert_buffer e1/Z class/AN2P
Information: Buffering net old_net. (HFS-701)
Error: library cell 'AN2P' is not a buffer or an inverter. (NLE-010)
```

The following example specifies a buffer for the library cell:

```
prompt> insert_buffer e1/Z class/B1I
Information: Buffering net old_net. (HFS-701)
Creating net 'eco_net' in design 'top'.
Creating cell 'eco_cell' in design 'top'.
Disconnecting net 'old_net' from pin 'e3/A'.
Disconnecting net 'old_net' from pin 'e4/A'.
Disconnecting net 'old_net' from pin 'e5/A'.
Connecting net 'eco_net' to pin 'e3/A'.
Connecting net 'eco_net' to pin 'e4/A'.
Connecting net 'eco_net' to pin 'e5/A'.
Connecting net 'eco_net' to pin 'eco_cell/Z'.
Connecting net 'old_net' to pin 'eco_cell/I'.
{eco_cell}
```

The following example specifies an inverter for the library cell. The command succeeds and creates the new cells named *eco_cell* and *eco_cell_1*. The new nets are named *eco_net* and *eco_net_1*.

```
prompt> insert_buffer e3/A class/IV
Information: Buffering net old_net. (HFS-701)
Creating net 'eco_net' in design 'top'.
Creating net 'eco_net_1' in design 'top'.
Creating cell 'eco_cell' in design 'top'.
Creating cell 'eco_cell_1' in design 'top'.
Disconnecting net 'old_net' from pin 'e3/A'.
Connecting net 'eco_net' to pin 'e3/A'.
Connecting net 'eco_net' to pin 'eco_cell/Z'.
Connecting net 'eco_net_1' to pin 'eco_cell/A'.
Connecting net 'eco_net_1' to pin 'eco_cell_1/Z'.
Connecting net 'old_net' to pin 'eco_cell_1/A'.
{eco_cell eco_cell_1}
```

```
prompt> report_cell-connections e3
*****
```

```
Report : cell
        -connections
*****
```

Connections for cell 'e3':

Reference: B1I
Library: class

Input Pins	Net
A	eco_net

Output Pins	Net
Z	out2

1
prompt> **report_net -connections eco_net**

Report : net
-connections

Connections for net 'eco_net':

Driver Pins	Type
eco_cell/Z	Output Pin (IV)

Load Pins	Type
e3/A	Input Pin (B1I)

1

SEE ALSO

- current_instance(2)
- get_pins(2)
- remove_buffer(2)
- report_cell(2)
- report_net(2)
- size_cell(2)
- PARA-060(n)

insert_clock_gating

Performs clock gating on an appropriately-prepared GTECH netlist.

SYNTAX

```
status insert_clock_gating  
  [-regular_only]  
  [-global]  
  [-no_hier]
```

ARGUMENTS

-regular_only

Disables enhanced clock gating during register-based clock gate insertion.

-global

Performs hierarchical clock gating on all subdesigns as one global step.

-no_hier

Limits clock gating to the top level of the current design.

DESCRIPTION

This command performs clock gating on an appropriately-prepared GTECH netlist. To insert or optimize clock gating on a mapped netlist, use the **compile** or **compile_ultra** command with the **-gate_clock** option. You can optionally use the **set_clock_gating_style** command beforehand to specify the structure of the clock-gating circuitry to be inserted. If no style is explicitly set, a default clock-gating style is applied. See the **set_clock_gating_style** command man page for more details on the default clock-gating style.

To control the names of modules, cells and gated clock nets created by this command, you can set different naming style variables listed at the end.

In addition to standard bank clock gating, the **insert_clock_gating** command searches for more clock gating opportunities by combining different register banks that share common enable conditions. This type of enhanced clock gating can result in extra clock-gating cells and/or gated registers.

The **insert_clock_gating** command performs clock gating at all levels of hierarchy in the design. By using the **-no_hier** option, the clock gating is limited to only the top level. If you want to perform clock gating on some, but not all, levels of hierarchy, the other levels can be excluded by setting **dont_touch** attributes on those designs or instances.

When the **-no_hier** option is omitted, clock gating is performed on all subdesigns. Every subdesign is processed independently. With the **-global** option, the design and its subdesigns are processed in one single global step. The advantage of the one-step hierarchical clock gating is twofold. First, the subdesigns are analyzed in their context, which is similar to boundary optimization during **compile** command. Second, the clock gates can be inserted at higher levels in the design, allowing for sharing of registers among different subdesigns. When a clock gate is inserted in a different hierarchical level than the registers that are being gated, new ports will be added to connect the gated clock and enable nets.

Important: when using the **-global** option, the operation is no longer context independent. The functionality of the subdesigns instantiated in the design can be modified in context of the current design. The functionality of the top level (current_design) is preserved. However, when certain subdesigns are also instantiated elsewhere (in a design different from the current design), the functionality of those designs can change. Therefore, it is not recommended to change the current design to that of a lower level and then run **insert_clock_gating -global**. If you choose to do so, you must uniquify the original design before changing the current design and running the **insert_clock_gating -global** command on a subdesign to guaranteed that the functionality of the original design is not altered.

During test (scan shifting), the clock gate has to be surpassed so that each clock pulse is passed onto the register. This is facilitated with a control-point insertion. See the **set_clock_gating_style** man page for details. If the clock gates have control points, all of them must be hooked up correctly to the top level test_mode or scan_enable pins using the **insert_dft** command.

EXAMPLES

The commands in the following example show the typical flow for using the **insert_clock_gating** command. After reading or elaborating the design and defining the clock ports, you set the clock-gating style to specify how to perform clock gating. The **insert_clock_gating** command inserts clock gating for the registers.

```
prompt> read_verilog design.v
prompt> current_design top
prompt> link
prompt> set_clock_gating_style -sequential latch
prompt> create_clock clk
prompt> insert_clock_gating
prompt> propagate_constraints -gate_clock
prompt> compile
```

The following example specifies using an integrated cell for the clock-gating circuitry. The **target_library** environment variable also contains the name of the library that contains the integrated cell. This example results in a mapped netlist that has integrated clock-gating cells.

```
prompt> set_clock_gating_style -sequential latch \
    -pos integrated -neg integrated -control_point before
prompt> set target_library {$target_library integrated_cell_library.db}
prompt> read_verilog design.v
prompt> current_design top
prompt> link
prompt> insert_clock_gating
prompt> uniquify
prompt> propagate_constraints -gate_clock
prompt> compile
```

SEE ALSO

compile(2)
compile_ultra(2)
create_clock(2)
insert_dft(2)
propagate_constraints(2)
set_clock_gating_registers(2)
set_clock_gating_style(2)
uniquify(2)
power_cg_gated_clock_net_naming_style(3)
power_cg_cell_naming_style(3)
power_cg_module_naming_style(3)
target_library(3)

insert_dft

Inserts DFT logic in the current design.

SYNTAX

status **insert_dft**

ARGUMENTS

The **insert_dft** command has no arguments.

DESCRIPTION

This command inserts DFT logic in the current design. The following types of DFT logic are supported:

- Standard scan chains
- Compressed scan chains with scan compression codec logic
- Clock-gating cell test pin connections
- AutoFix logic
- Test points
- On-chip clock (OCC) controller blocks
- Core wrappers
- ANSI/IEEE Std 1149.1/1149.6-compliant boundary-scan logic

By default, **insert_dft** performs standard scan chain insertion and routing. To insert compressed scan, boundary scan, or other types of DFT logic, use the applicable DFT configuration commands for that DFT logic type before running **insert_dft**.

The **insert_dft** command is supported with multicorner-multimode (MCMM) technology, which is available with Design Compiler Graphical.

Use the **set_dft_location** command to specify an alternate user-defined location for one or more types of DFT logic. For more information, see the man page.

Scan Synthesis Description

Scan cells and DFT logic cells are chosen from the technology library specified in the **target_library** variable. You must define a target library before issuing the **insert_dft** command.

The **insert_dft** command supports top-down, bottom-up, and middle-out scan insertion methodologies. In the process of adding DFT logic, testable versions of the subdesigns and top-level design are created. For testable subdesigns, new designs are created in the database that are distinct from the original subdesigns. The new subdesigns are named according to the value of the **insert_test_design_naming_style** variable. Note that the name of the current design is not changed during the **insert_dft** execution.

If existing test ports have been identified with the **set_dft_signal**, **insert_dft** will make connections to these ports.

If test ports have not been identified, new ports are added to the design as needed. The new ports are named according to the following variables:

```
test_mode_port_naming_style
test_mode_port_inverted_naming_style
test_clock_port_naming_style
test_scan_clock_a_port_naming_style
test_scan_clock_b_port_naming_style
test_scan_clock_port_naming_style
test_scan_enable_inverted_port_naming_style
test_scan_enable_port_naming_style
test_scan_in_port_naming_style
test_scan_out_port_naming_style
```

These variables are described in their respective man pages.

The DFT insertion process performed by the **insert_dft** command works as described below.

First, existing logic is marked so scan insertion can detect whether a particular piece of logic has been added.

If test DRC results from a previous **dft_drc** command are not available, **insert_dft** invokes test DRC for the specified scan style. DRC generates the clock information that scan insertion retrieves. DRC also marks violating cells that cannot belong to scan chains.

Next, **insert_dft** architects scan chains into the design. The **insert_dft** command applies a scan-equivalence process to nonscan cells that passed DRC. Scan equivalents are initially selected based on library function identifiers. If no scan equivalent for a sequential cell is found using this behavior, **insert_dft** uses constraint-based sequential mapping techniques. These techniques consider the logic function implemented by each flip-flop instance to be scan replaced, along with immediately surrounding logic. This consideration generates a set of possible alternatives, from which the best replacement is selected based on **scan_style**, design rule considerations (for example, **connection_class** and **max_fanout**), and design area.

Use the **preview_dft** command to view the proposed architecture without inserting it. By default, **insert_dft** constructs as many scan chains as there are clocks and edges. By setting the **-clock_mixing** option of **set_scan_configuration**, the number of scan chains created based on clocks can be controlled. Set the **-clock_mixing** option to **no_mix**, **mix_edges**, or **mix_clocks**. If you want **insert_dft** to build more than the minimal scan chain count, set the **-chain_count** option of **set_scan_configuration** command. Multiple scan chains are also constructed to reflect the assignment of scan cells to scan chains by using the **set_scan_path** command.

When **insert_dft** constructs multiple scan chains, scan cells are allocated to scan chains based on the following criteria:

- The **set_scan_path** command
- Clock signals in the design
- Subdesign scan chains
- Design hierarchy
- Names of individual scan cells.

Scan cells are ordered on scan chains based on the following criteria:

- The **set_scan_path** command position
- Scan
- Clock trigger times
- Scan clock domains
- Scan cell names

The **insert_dft** command overrides the **set_scan_path** command positions to ensure functional chains if **set_scan_path** is not inserting lock-up latches.

Next, the **insert_dft** command makes any needed clock-gating cell test pin connections. These test pin connections place the clock-gating cells into a bypass mode when the test pin is asserted. For information on the available controls for these test pin connections, see the man pages for the **set_dft_clock_gating_configuration** and **set_dft_clock_gating_pin** commands.

insert_dft ensures that the internal three-state buses, which are on-chip buses not driven by pad cells, are disabled during scan shift. It also ensures that bidirectional ports are properly configured during scan shift. You can force the **insert_dft** command to disregard these steps by using the **-fix_bus disable** or **-fix_bidirectional disable** options of the **set_dft_configuration** command.

The **insert_dft** command avoids adding unnecessary logic by ignoring buses that already have exactly one active driver during scan shift, and bidirectional ports that are properly configured for scan shift. It takes into account any constant signals that have been applied with the **set_dft_signal -view existing_dft -type Constant** command.

Having identified candidate buses, **insert_dft** identifies the disabling logic required. The command iterates through all three-state bus drivers and computes the logic values that must be applied to input pins to disable the bus. It does not succeed in the following situations, when:

- A bus driver belongs to a design that has a **dont_touch** attribute.
- Any of the three-state drivers cannot be disabled.
- Conflicting logic values are required to disable three-state drivers.
- Disabling values conflict with those of previous buses.

The **insert_dft** command then adds generic disabling logic where necessary. It finds and gates all pins that do not hold the values they require during scan shift.

The command processes all bidirectional ports. It discards the following:

- Ports used to apply scan enable signals to the design
- Degenerated bidirectional ports that are configured as inputs or outputs by constant logic values
- Ports that do not have exactly 1 three-state driver that is not in a **dont_touch** design

The **insert_dft** command establishes the direction to which bidirectional ports must be configured during scan shift. Scan outputs are turned outwards, and other scan signals are turned inwards. By default, all other bidirectional ports are turned inwards. To turn bidirectional ports outwards, use the **set_autofix_element** command with the **-type external_bus** and **-method output** options.

The **insert_dft** command computes the logic values that must be applied to the driver input pins to configure each port. It fails if required logic values conflict with previous logic values. The command then adds generic configuration logic where necessary. It finds all pins that do not hold the required values during scan shift, and inserts appropriate test mode logic.

Having disabled three-state buses and configured bidirectional ports, **insert_dft** builds the scan chains if the **set_scan_configuration** command has not been previously specified with the **-route false** option. For each chain, **insert_dft** determines the scan-in pin. It first looks for **ScanDataIn** signals specified by using the **set_dft_signal** commands. It then looks for

design ports with **ScanDataIn** **signal_type** attributes. If it finds no such ports, it creates a dedicated scan-in port for the chain.

Having identified the scan-in pin, **insert_dft** jumps over the connected pad if it is not in a **dont_touch** design. It first tries finding a true or inverting path through the pad that is sensitized by test hold, scan enable, and bidirectional control signals, and does not need any additional disabling. If not successful, it gets the vector required to enable a path through the pad and the resulting hookup pin. It finds all pins that do not hold the necessary values during scan shift and inserts appropriate test mode logic. Note that **insert_dft** does not have to add three-state disabling logic to ensure that the pad is in input mode because this was performed during a previous step.

The **insert_dft** command then jumps through input buffers. Starting at the load pin, it jumps over single-fanout inverters and buffers to find a better hookup pin. The scan sense is toggled as this process jumps over inverters.

Given the best hookup pin, scan insertion builds the rest of the chain. The preferred routing order, specified by using the **set_scan_path** command, is used to route between serial signals and scan cells. If no preferred order is specified, it uses a default ordering scheme.

If the **set_scan_configuration** command has not been previously executed with the **-add_lockup false** option, the **insert_dft** command inserts lock-up latches at clock-domain boundaries. This compensates for clock skew and ensures a glitch-free scan shift. You can specify to **insert_dft** to add lock-up latches at the end of chain by using the **set_scan_configuration** command with **-insert_terminal_lockup**.

The **insert_dft** command can reuse functionally-connected **ScanDataIn** ports or pins for scan chain routing. It does not recognize a pin as functionally connected if the following conditions are true:

- The pin is not connected to a net.
- The pin has been created by **insert_dft**.
- The pin is a driver and does not have a load, or the pin is connected to a subdesign output port that **insert_dft** created.
- The pin is a load and does not have a driver, or the pin is connected to a subdesign input port that **insert_dft** created.

The **insert_dft** command does not touch optional methodology signal pins that are functionally connected. It reconnects mandatory methodology signal pins that are functionally connected. A warning message appears in this case.

The **insert_dft** command selects the scan-out driver for each cell by using the following criteria:

- It identifies the scan-out signal by looking at **signal_type** attributes. If pins with both **ScanDataIn** and **ScanDataIn** with inverted attributes are present, it chooses the noninverted pin, unless the other has a routing position.
- It considers all of the equal and opposite output pins (such as Q and QN) on the final scan cell in the chain, and chooses the output pin with the most slack. If equal, it chooses the driver with greatest drive strength.

You can set the **test_disable_find_best_scan_out** environment variable to **true** to disable this behavior.

To finish construction, **insert_dft** locates the scan-out port for the chain. If a scan-out port has not been previously specified with the **set_dft_signal** command, **insert_dft** creates a new one. If an existing scan-out port is found, **insert_dft** analyzes it to avoid inserting redundant multiplexing logic. It first establishes whether there is already a signal path between the last scan-out pin and the scan-out port when scan enable, test hold, and bidirectional control conditions are asserted. If the scan out port is connected to a three-state driver or a pad, it uses the reverse of the method described earlier for input ports to jump back through three-state drivers, pads, buffers, or inverters to a hookup pin. If the this pin is already functionally connected, **insert_dft** adds multiplexing logic and connects the selected output pin of the chain's last scan cell to the multiplexer input. If the scan-out net connects to some other existing port, **insert_dft** isolates the scan-out port connection by making the connection with a buffer. The reason for this port isolation is because some downstream tools cannot handle nets that fanout to multiple ports.

Finally, **insert_dft** routes global signals (including any scan enable and test clock signals) and performs a default technology mapping for all new generic logic (including disabling logic and multiplexers). It introduces dedicated test clock signals for the **clocked_scan**, **lssd**, and **scan_enabled_lssd** scan styles.

When **set_dft_configuration -scan disable -clock_gating enable** has been previously specified, **insert_dft** performs only clock-

gating cell routing. In this case, the **create_test_protocol** command should be run after **insert_dft** to update the CTL model with the correct ScanEnable or TestMode signal assertions.

Typically, at this point, the **insert_dft** command has violated compile design rules and constraints. If the **set_dft_insertion_configuration-synthesis_optimization none** command has been previously issued, or if Design Compiler topographical mode is used, optimization is skipped and the initial mapping is retained. Otherwise, **insert_dft** optimizes the design according to the setting of the **-map_effort** option of the **set_dft_insertion_configuration** command. For more information, see the **set_dft_insertion_configuration** man page. Note that **insert_dft** does not optimize the ScanEnable net to meet timing constraints.

In Design Compiler topographical mode, you should follow DFT insertion with an incremental topographical compile.

At this point, some cleanup is necessary. The **insert_dft** command resolves connection class violations, replaces and merges logic constants, then uniquifies the design. This is because you might want to write out subdesigns that **insert_dft** has modified as instances of new designs. In some cases, where test modifies instances of the same design differently, **insert_dft** produces more than one new design. It uses a technique called selective uniquification to produce new designs only when necessary. This technique generates a canonical netlist ID that does not depend on net-names or cell-names. The canonical netlist ID works only on gate-type, net connections, and port-names.

In the case of a non-uniquified hierarchical design, forced uniquification is performed to record the changes to the design. This is because you might want to write out subdesigns that **insert_dft** has modified as instances of new designs. In some cases, where test modifies instances of the same design differently, **insert_dft** produces more than one new design. You might want **insert_dft** to do this only when necessary.

The **insert_dft** command only considers **dont_touch** attributes for basic initial scan replacement of nonscan cells, such as those that would occur if a test-ready compile was not used. They are ignored for scan stitching, test signal routing, logic insertion, and identified shift register splitting (which scan-replaces the new head flip-flop). To control the scope of DFT insertion, use test-specific directives such as **set_scan_element false** or **set_scan_configuration-exclude_elements**.

BSD Synthesis Description

The **insert_dft** command synthesizes ANSI/IEEE Std 1149.1/1149.6-compliant boundary-scan logic using DesignWare macro cells when you specify the **-bsd enable** option of the **set_dft_configuration** command.

Synthesis is done at the gate level. The newly-synthesized BSD logic is mapped to the target library. The BSR cells added during BSD insertion are not uniquified. You must run the **uniquify** command after BSD insertion if you need a uniquified design. The **insert_dft** command can use a design with or without core logic, but pad cells must be present on the top-level design ports.

All JTAG test access ports (TAPs) must be specified. If you omit the JTRST signal using the **set_bsd_configuration -asynchronous_reset false** command, you must specify the remaining four mandatory ports. otherwise, you must specify all five ports.

Depending on the instructions you selected to implement with the **set_bsd_instruction** command, by default the instruction register (IR) is synthesized with the minimum number of bits to accommodate the set of instructions. However, you can enforce one-hot encoding of the instruction register by using **set_bsd_configuration-instruction_encoding one_hot**. The **insert_dft** command synthesizes BSR cells using DesignWare BC_1 on all the design output ports except on bidirectional ports, where a BC_7 is synthesized. On all input ports and for all control points, BC_2 BSR cells are synthesized. If you set system clocks, a BC_4 is synthesized on these design ports. If you used some design ports other than the actual controlling ports to control the pads' controlling logic (for example, an extra controlling port to disable all three-state outputs together), logic is synthesized to allow the actual controlling signal to do the controlling during the boundary-scan testing. This removes the need for subsequently setting the extra controlling ports as compliance-enable ports.

It is not possible to synthesize boundary scan logic without pads. If the design does not have pads, the command terminates.

Error messages can result from missing TAP ports, missing pad cells or missing pad cell functionality in the library. A message is printed indicating an inability to proceed, and the command exits.

By default, synthesis of boundary scan logic is done in compliance with the IEEE1149.1-2001 standard. You can switch back to the IEEE1149.1-1993 standard by setting the **-std {ieee1149.1_1993}** option of the **set_bsd_configuration** command. To

implement the IEEE1149.6-2003 standard, specify the **-std {ieee1149.6_2003}** option of the **set_bsd_configuration** command.

By default, synthesis of boundary scan logic is done using the new DesignWare tap, DW_tap_uc. To use the old DesignWare tap, DW_tap, instead, set the **bsd_use_old_tap** variable to true.

To use user-defined BSR/TAP designs in BSD synthesis, use the **define_dft_design** command to specify the user-defined designs.

For more details on the IEEE Std 1149.1/1149.6 rules, see the *IEEE Std Test Access Port and Boundary-Scan Architecture*.

EXAMPLES

The following example shows the usage syntax for the **insert_dft** command on the design **WC66** without fixing constraint violations and compile design rules:

```
prompt> current_design WC66
```

```
prompt> insert_dft
```

The following command performs synthesis of boundary-scan logic in compliance with ANSI/IEEE Std 1149.1 on the current design, and illustrates the types of messages that are output:

```
prompt> current_design M1
prompt> set_dft_configuration -scan disable -bsd enable
prompt> set_dft_signal-type tdi -port my_tdi
prompt> set_dft_signal-type tck -port my_tck
prompt> set_dft_signal-type tms -port my_tms
prompt> set_dft_signal-type trst -port my_trst
prompt> set_dft_signal-type tdo -port my_tdo
prompt> create_clock -p 10 my_system_clock
prompt> insert_dft
```

```
Loading design 'M1'
```

```
...
```

```
Generating the TAP
```

```
...
```

```
Generating the BSR cells
```

```
...
```

```
Generating the control logic
```

```
Writing implemented instructions information to the design
```

```
Creating Hierarchy for Boundary Scan Logic ...
```

```
...
```

```
1
```

SEE ALSO

[create_test_protocol\(2\)](#)

[current_design\(2\)](#)

[dft_drc\(2\)](#)

[preview_dft\(2\)](#)

[set_dft_configuration\(2\)](#)

```
set_dft_insertion_configuration(2)
set_dft_location(2)
set_scan_configuration(2)
```

insert_isolation_cell

Inserts isolation cells on the specified nets, pins or ports. Isolation cell is a general term that applies to isolation cells and enabled level-shifter cells.

SYNTAX

```
status insert_isolation_cell
  [-force]
  [-verbose]
  -enable enable_signal
  -object_list objects
  -reference lib_cell_name
```

Data Types

```
enable_signal  collection
objects        collection
lib_cell_name  string
```

ARGUMENTS

-force

Forces insertion of isolation cells on nets with the **dont_touch** or **always_on** attributes.

-verbose

Provides a detailed report on the isolation cells inserted in the design.

-enable *enable_signal*

Specifies the signal in the design to be connected to the enable pin of the inserted isolation cells. You can specify this signal as either a net, a pin, or a port.

This option is required.

-object_list *objects*

Specifies a list of pins, ports, or nets on which to insert the isolation cells. If you specify object names, the tool resolves conflicts in the order of ports, pins and nets.

This option is required.

-reference *lib_cell_name*

Specifies the name of the library cell of the isolation cell. This can be either an isolation cell or an enabled level-shifter cell.

This option is required.

DESCRIPTION

The **insert_isolation_cell** command inserts isolation cells on the netlist objects specified in the **-object_list** option. It connects the enable pin of each inserted cell to the enable signal specified in the **-enable** option. The tool determines the set of possible reference cells for the isolation or enabled level-shifter cells by searching the link libraries for cells matching the name specified in the **-reference** option.

The tool selects the reference cell for each netlist object by using the following strategy:

- If the driver and all loads of the associated net operate at the same voltage, the tool selects an isolation cell that matches the operating condition of the subdesign in which the net resides. If such an isolation cell is not found, but an appropriate enabled level-shifter cell is found, the tool selects the enabled level-shifter cell.
- If the driver and loads of the associated net operate at different voltages (a multivoltage net), the tool selects the first enabled level-shifter cell that implements voltage-level shifting. Isolation cells are not used for multivoltage nets.

The tool determines where to put each isolation or enabled level-shifter cell based on the type of the netlist object:

- If you specify a subdesign port, the tool inserts the isolation or enabled level-shifter cell on the net connection inside the subdesign, next to the specified port. When you specify a driver port, the isolation or enabled level-shifter cell connects to all paths driven by the port. When you specify a load port, the isolation or enabled level-shifter cell connects only to the path ending at the load port.

To specify subdesign ports, use the **get_ports** command.

- If you specify a subdesign pin, the tool inserts the isolation or enabled level-shifter cell on the net connection outside the subdesign, next to the specified pin. When you specify a driver pin, the isolation or enabled level-shifter cell connects to all paths driven by the pin. When you specify a load pin, the isolation or enabled level-shifter cell connects only to the path ending at the load pin.

To specify subdesign pins, use the **get_pins** command.

- If you specify a net, the tool inserts the isolation or enabled level-shifter cell in net's parent subdesign.

To specify a net, use the **get_nets** command.

By default, the tool uses the <power_domain_name>_ISO_<count> naming rule to generate names for isolation cells. Each isolation cell inside its own domain gets a unique count. You can add a prefix string to the automatically-generated name by specifying a value for the **isolation_cell_naming_prefix** variable.

In the following situations, the command does not insert an isolation or enabled level-shifter cell on the associated net and flags the net as a violating net:

- The tool cannot find an appropriate library cell for the net.
- The net has multiple drivers.
- The net is connected to a bidirectional pin.
- The net has a **dont_touch** or **always_on** attribute. You can use the **-force** option to force the insertion of isolation cells on such nets. However, do not use this option when the operating conditions of the isolation cell and the subdesign do not match.

Note that this command is supported only on unqualified designs and designs with power domains.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example inserts a GTECH isolation cell on the *IN* net inside the *I1* hierarchical instance. The enable pin of the isolation cell is wired to the *ENABLE* port of the design.

```
prompt> insert_isolation_cell -reference GTECH_ISO1_EN0 \  
-object [get_nets {I1/IN}] \  
-enable [get_ports ENABLE]
```

The following example inserts a GTECH isolation cell on the net connected to port *A* of instance *I1*. Because a port is specified, the isolation cell is inserted inside the *I1* block.

```
prompt> insert_isolation_cell -reference GTECH_ISO1_EN0 \  
-object [get_ports {I1/A}] \  
-enable [get_ports ENABLE]
```

The following example inserts a GTECH isolation cell on the net connected to pin *A* of instance *I1*. Because a pin is specified, the isolation cell is inserted outside the *I1* block.

```
prompt> insert_isolation_cell -reference GTECH_ISO1_EN0 \  
-object [get_pins {I1/A}] \  
-enable [get_ports ENABLE]
```

The following example inserts an enabled level shifter on the *RET* net. The enable pin of the enabled level shifter is wired to the *en* net inside the *PWRCTRL* hierarchical instance.

```
prompt> insert_isolation_cell -reference LVLHLEHX2 \  
-object [get_nets RET] \  
-enable [get_nets PWRCTRL/en]
```

SEE ALSO

get_nets(2)
get_pins(2)
get_ports(2)
remove_isolation_cell(2)

insert_mv_cells

Inserts isolation and level-shifter cells to the design.

SYNTAX

```
status insert_mv_cells
  [-isolation]
  [-level_shifter]
  [-all]
  [-verbose]
```

ARGUMENTS

-isolation

Inserts isolation cells.

-level_shifter

Inserts level-shifter cells.

-all

Inserts both isolation and level-shifter cells. This is the default behavior.

-verbose

Inserts the isolation cells in verbose mode. It does not affect the actual insertion but this option might give additional information if level-shifters were not inserted. It does not provide any additional information regarding isolation-cell insertion.

DESCRIPTION

This command inserts isolation cells and level-shifter cells as specified by the UPF specification. It allows these cells to be inserted without having to run a full optimization on the design using the **compile_ultra** command. The result might not be identical to that of automatic insertion of these cells during the **compile_ultra** activity. This difference is due to additional optimizations done to the design during **compile_ultra** activity, before the insertion of isolation and level-shifter cells. All UPF strategies are applied for isolation and level-shifter cells. It is not possible to select specific strategies. If there are repeater supply attributes on the domain boundaries, repeaters are inserted before the insertion of other power management cells.

EXAMPLES

The following example uses the **insert_mv_cells** command:

```
prompt> insert_mv_cells
```

SEE ALSO

- compile_ultra(2)
- set_isolation(2)
- set_isolation_control(2)
- set_level_shifter(2)
- set_port_attributes(2)

is_false

Tests the value of a specified variable, and returns 1 if the value is 0 or the case-insensitive string **false**; returns 0 if the value is 1 or the case-insensitive string **true**.

SYNTAX

```
status is_false  
    value
```

Data Types

value string

ARGUMENTS

value

Specifies the name of the variable whose value is to be tested.

DESCRIPTION

This command tests the value of a specified variable, and returns 1 if the value is either 0 or the case-insensitive string **false**. The command returns 0 if the value is either 1 or the case-insensitive string **true**. Any value other than 1, 0, **true**, or **false** generates a Tcl error message.

The **is_false** command is used in writing scripts that test Boolean variables that can be set to 1, 0, or the case-insensitive strings **true** or **false**. When such variables are set to **true** or **false**, they cannot be tested in the negative in an **if** statement by simple variable substitution, because they do not constitute a true or false condition. The following example is not legal Tcl:

```
set x FALSE  
if { !$x } {  
    set y TRUE  
}
```

This results in a Tcl error message, indicating that you cannot use a non-numeric string as the operand of "!". So, although you can test the positive condition, **is_false** allows you to test both conditions safely.

EXAMPLES

The following example shows the use of the **is_false** command:

```
prompt> set x TRUE  
TRUE  
  
prompt> if { ![is_false $x] } {  
?   set y TRUE  
?   }  
TRUE  
  
prompt>
```

SEE ALSO

expr(2)
if(2)
is_true(2)

is_true

Tests the value of a specified variable, and returns 1 if the value is 1 or the case-insensitive string **true**; returns 0 if the value is 0 or the case-insensitive string **false**.

SYNTAX

```
status is_true
      value
```

Data Types

value string

ARGUMENTS

value

Specifies the name of the variable whose value is to be tested.

DESCRIPTION

This command tests the value of a specified variable, and returns 1 if the value is either 1 or the case-insensitive string **true**. The command returns 0 if the value is either 0 or the case-insensitive string **false**. Any value other than 1, 0, **true**, or **false** generates a Tcl error message.

The **is_true** command is used in writing scripts that test Boolean variables that can be set to 1, 0, or the case-insensitive strings **true** or **false**. When such variables are set to **true** or **false**, they cannot be tested in the negative in an **if** statement by simple variable substitution, because they do not constitute a true or false condition. The following example is not legal Tcl:

```
set x TRUE
if { !$x } {
    set y FALSE
}
```

This results in a Tcl error message, indicating that you cannot use a non-numeric string as the operand of "!". So, although you can test the positive condition, **is_true** allows you to test both conditions safely.

EXAMPLES

The following example shows the use of the **is_true** command:

```
prompt> set x FALSE  
FALSE
```

```
prompt> if { ![is_true $x] } {  
?   set y FALSE  
?   }
```

```
FALSE
```

```
prompt>
```

SEE ALSO

```
expr(2)  
if(2)  
is_false(2)
```

lib2saif

Creates a forward-annotation SAIF file for a specified technology library.

SYNTAX

```
status lib2saif
  [-output file_name]
  library
  [-lib_pathname lib_path_name]
```

Data Types

<i>file_name</i>	string
<i>library</i>	list
<i>lib_path_name</i>	list

ARGUMENTS

-output *file_name*

Specifies the name of the library forward-annotation SAIF file created by **lib2saif**. The default file name is the value of the **power_sdpgd_saif_file** variable. By default, the value of the **power_sdpgd_saif_file** variable is *power_sdpgd.saif*.

library

Specifies the library name, or library file name for which the forward SAIF file is generated. If a library file name is specified then the library must be in .db format. The library must have state-dependent leakage power characterization and/or pin-level state-dependent and/or path-dependent internal power characterization for a library forward SAIF file to be generated.

-lib_pathname *lib_path_name*

Specifies the path name to the simulation library information. This is optional, and is only required if the library is not in the simulation tool's current path during simulation.

DESCRIPTION

This command creates a forward-annotation SAIF file that contains the state-dependent and path-dependent information for library cells. The library forward-annotation SAIF file is used in flows where a gate-level back-annotation SAIF file with state-dependent and/or path-dependent switching activity is generated.

For details on the SAIF format, see the SAIF specification.

For details on state-dependent and path-dependent power characterization, see the Library Compiler and Power Compiler documentation.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples illustrate the use of this command:

```
prompt> lib2saif -out a130.saif asic130_typical
```

```
prompt> lib2saif -out a130.saif ../libs/asic130.db
```

SEE ALSO

report_power(2)
set_switching_activity(2)

license_users

Lists the current users of the Synopsys licensed features.

SYNTAX

```
status license_users
[feature_list]
```

Data Types

feature_list list

ARGUMENTS

feature_list

Lists the licensed features for which to obtain the information. If more than one feature is specified, they must be enclosed in braces ({}). See the Synopsys *Installation Guide: UNIX-Based Platforms* for a list of features supported by the current release, or determine from the key file all of the features that are licensed at your site.

DESCRIPTION

This command displays information about all of the licenses, related users, and host names currently in use. If a feature list is specified, only information about those features is displayed.

The **license_users** command is valid only when Network Licensing is enabled.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In this example, all of the users of the Synopsys features are displayed:

```
prompt> license_users
```

```
krig@node1 DC-Expert, Design-Compiler, Design-Vision,
```

```
    Milkyway-Interface
doris@node2  Design-Compiler, HDL-Compiler, Power-Optimization
schen@node3  DC-Expert, DC-Ultra-Features, DC-Ultra-Opt,
    Design-Compiler,
wru@node4    DC-Expert, DC-Ultra-Features, DC-Ultra-Opt,
    Design-Compiler, DesignWare,
    Milkyway-Interface, Test-Compiler,
    VHDL-Compiler
```

4 users listed.

This example shows the users of the "HDL-Compiler" or "VHDL-Compiler" features.

```
prompt> license_users {HDL-Compiler VHDL-Compiler}
```

```
doris@node2  Design-Compiler, HDL-Compiler, Power-Optimization
wru@node4    DC-Expert, DC-Ultra-Features, DC-Ultra-Opt,
    Design-Compiler, DesignWare,
    Milkyway-Interface, Test-Compiler,
    VHDL-Compiler
```

2 users listed.

SEE ALSO

```
get_license(2)
remove_license(2)
```

link

Resolves design references.

SYNTAX

status **link**

ARGUMENTS

The **link** command has no arguments.

DESCRIPTION

Performs a name-based resolution of design references for the current design. For a design to be complete, it needs to be connected to all of the library components and designs it references. The references must be located and linked to the current design in order for the design to be functional. The purpose of this command is to locate all of the designs and library components referenced in the current design and connect (link) them to the current design.

The **link** command uses the **link_library** and **search_path** variables along with the **local_link_library** design attribute to resolve design references. The **local_link_library** attribute and **link_library** variable specify a list of design and library files. A "*" entry in the value of the **link_library** variable indicates that **link** should search all the designs already loaded in memory. If the **link_library** variable has no "*" entry, the already-loaded designs are not searched. The default value for the **link_library** variable is "your_library.db". The **search_path** variable specifies a list of directory names that the **link** command uses to search for **link_library** or **local_link_library** files.

If the reference is to a parameterized design (and the design is not already in memory), **link** automatically builds the template with the specified parameters. Parameterized designs can only be specified using HDL code.

The **link** command searches for referenced designs by looking in each file specified by the **local_link_library** attribute and the **link_library** variable. If the **local_link_library** attribute is set on the current design, those files are searched ahead of those specified by the **link_library** variable. For simple file names (names that contain no '/' character), the **link** command looks for the files in the directories specified by the **search_path** variable. For absolute or relative path names, the **search_path** variable is not used. If the referenced designs are not found in any of the specified files, the **link** command looks in the search path directories for any .ddc or .db files that have the same name as the referenced design (for example, adder.ddc or adder.db). In this case, the .ddc files are searched first.

The order of directories in the search path and of the files in the link libraries is important. Search order during a link is

1. Local link library files
2. Link library files

3. Search path directories

The first occurrence of a design reference is used.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

In all modes, if a design named TOP refers to design named test in its implementation, the **link** command must find and connect the test design to this reference. The **link** command first searches the files specified in the link libraries for TOP. If it does not find the test design in the link libraries, it searches the directories specified in the **search_path** variable for a file named test.ddc or test.db. If it does not find it there, a warning is issued stating that the reference cannot be resolved.

Assume that the **local_link_library** attribute, **search_path** variable, and **link_library** variable for the previous example are set as follows:

```
prompt> set_app_var search_path "$synopsys_root/libraries /usr/bill/project"
/usr/synopsys/libraries /usr/bill/project
```

```
prompt> set_local_link_library tech1_lib.db
1
```

```
prompt> set_app_var link_library "** tech2_lib.db ./project.db"
* tech2_lib.db ./project.db
```

To find the test design, the **link** command performs the following sequence:

Step 1.

Read the tech1_lib.db file. Because this is a simple filename, the search path is used to find the file's location.

Step 2.

Search all the designs already loaded in memory for a design called test.

Step 3.

Read the tech2_lib.db file. Because this is a simple filename, the search path is used to find the file's location.

Step 4.

Read the project.db file from the current directory. Because this is a complex filename (the current directory is specified), the search path is not used to find the file.

Step 5.

Look for the test design in the list of designs contained in tech1_lib.db, tech2_lib.db, and project.db.

Step 6.

If the test design is not found in Step 5, search for a file named either test.ddc or test.db using the search path. In the unlikely event that test.ddc and test.db both exist, the .ddc file is used.

The following examples demonstrate the different ways the **link** command can search for referenced designs.

Example 1

The **search_path** variable is set to a list of directories where designs and technology libraries reside. In the following example, the top, counter, and controller designs are built using components from the tech_lib library. You need to read in only the top-level design. The **link** command automatically loads designs from both the link library files and, as needed, files in the search path.

```
prompt> set_app_var search_path "$synopsys_root/libraries /usr/bill"
/usr/synopsys/libraries /usr/bill
```

```
prompt> read_file top.ddc
Loading db file '/usr/synopsys/libraries/syn/gtech.db'
Loading db file '/usr/synopsys/libraries/syn/standard.sldb'
Loading link library 'gtech'
Reading ddc file '/usr/bill/top.ddc'.
Loaded 1 design.
Current design is 'top'.
{top}
```

```
prompt> set_app_var link_library "" tech_lib.db"
* tech_lib.db
```

```
prompt> link
```

```
Linking design 'top'
Using the following designs and libraries:
```

```
-----
top                /usr/bill/top.ddc
tech_lib (library) /usr/synopsys/libraries/tech_lib.db
```

```
Reading ddc file '/usr/bill/controller.ddc'.
Reading ddc file '/usr/bill/counter.ddc'.
1
```

Example 2

The following example shows that reading a design into memory has no effect on the **link** command if the "" is missing from the **link_library** variable. The adder design is read into memory explicitly, but cannot be found on the search path or in the link libraries. Therefore, it is unresolved during linking.

This command sets the **search_path** variable without including the directory that contains the adder.ddc file:

```
prompt> set_app_var search_path "$synopsys_root/libraries /usr/bill"
/usr/synopsys/libraries /usr/bill
```

The next two commands read the adder design into memory and set the current design to top:

```
prompt> read_file /usr/bill/dc/adder.ddc
Loading db file '/usr/synopsys/libraries/syn/gtech.db'
Loading db file '/usr/synopsys/libraries/syn/standard.sldb'
Loading link library 'gtech'
Reading ddc file '/usr/bill/dc/adder.ddc'.
Loaded 3 designs.
Current design is 'adder'.
adder mux81 top
```

```
prompt> current_design top
```

Current design is 'top'.
{top}

The next command sets the **link_library** variable without including the adder.ddc file or **""**:

```
prompt> set_app_var link_library "tech_lib.db"
tech_lib.db
```

The following command shows that the **link** command fails for the designs in adder.ddc:

```
prompt> link

Linking design 'top'
Using the following designs and libraries:
-----
tech_lib (library)      /usr/synopsys/libraries/tech_lib.db

Warning: Unable to resolve reference 'adder' in 'top'. (LINK-5)
Warning: Unable to resolve reference 'mux81' in 'top'. (LINK-5)
```

Adding **""** to the **link_library** variable allows the design to link successfully:

```
prompt> set_app_var link_library "" $link_library"
* tech_lib.db

prompt> link

Linking design 'top'
Using the following designs and libraries:
-----
* (3 designs)          /usr/bill/dc/adder.ddc, etc
tech_lib (library)     /usr/synopsys/libraries/tech_lib.db

1
```

An alternate solution for the link failure is to add the adder.ddc file to the **link_library** variable:

```
prompt> set_app_var link_library "tech_lib.db /usr/bill/dc/adder.ddc"
tech_lib.db /usr/bill/dc/adder.ddc

prompt> link

Linking design 'top'
Using the following designs and libraries:
-----
* (3 designs)          /usr/bill/dc/adder.ddc, etc
tech_lib (library)     /usr/synopsys/libraries/tech_lib.db

1
```

Example 3

This example shows how to handle files with multiple designs. If you need to link with a file of multiple designs, put that file in your **link_library**.

If a design is not found in the **link_library** files, **link** searches the **search_path** directories for a file of the same name as the referenced design, with a .db or .ddc suffix. If your multiple-design file has a different file name then it will not be found, even if it is located on the **search_path**.

```
prompt> set_app_var search_path "$synopsys_root/libraries /usr/project"
/usr/synopsys/libraries /usr/project
```

The following command sets **link_library** without including "mylib.ddc":

```
prompt> set_app_var link_library "tech_lib.db"
tech_lib.db
```

The link fails for designs in "mylib" that are not found in any **link_library** files or in files in **search_path** directories that are named with the design names:

```
prompt> read_file top.ddc
Loading db file '/usr/synopsys/libraries/syn/gtech.db'
Loading db file '/usr/synopsys/libraries/syn/standard.sldb'
Loading link library 'gtech'
Reading ddc file '/usr/project/top.ddc'.
Loaded design.
Current design is 'top'.
top
```

```
prompt> link
```

```
Linking design 'top'
Using the following designs and libraries:
```

```
-----
tech_lib (library)      /usr/synopsys/libraries/tech_lib.db
```

```
Warning: Unable to resolve reference 'adder' in 'top'. (LINK-5)
```

```
Warning: Unable to resolve reference 'mux81' in 'top'. (LINK-5)
```

Because the adder and mux81 designs are in the mylib.ddc file, adding this file to the **link_library** variable enables a successful link:

```
prompt> lappend link_library mylib.ddc
tech_lib.db mylib.ddc
```

```
prompt> link
Reading ddc file '/usr/project/mylib.ddc'.
```

```
Linking design 'top'
Using the following designs and libraries:
```

```
-----
tech_lib (library)      /usr/synopsys/libraries/tech_lib.db
* (2 designs)          /usr/project/mylib.ddc, etc
```

1

SEE ALSO

```
current_design(2)
set_local_link_library(2)
list_designs(2)
which(2)
auto_link_options(3)
```

```
link_library(3)
search_path(3)
```

list

Creates a list.

SYNTAX

int **list** *arg1 arg2 ... argn*

ARGUMENTS

arg1 arg2 ... argn

Items to be included in the returned list.

DESCRIPTION

This command returns a list comprised of all the specified arguments or an empty string if no arguments are specified. Braces and backslashes are added as necessary, so that the **index** command can be used on the result to re-extract the original arguments, and also so that the **eval** command can be used to execute the resulting list, with *arg1* comprising the command's name and the other arguments comprising its arguments. Unlike the **concat** command, which removes one level of grouping before forming the list, the **list** command works directly from the original arguments. For example, the following command

```
list a b {c d e} {f {g h}}
```

returns

```
a b {c d e} {f {g h}}
```

while the **concat** command, issued with the same arguments, returns the following:

```
a b c d e f {g h}
```

SEE ALSO

help(2)

list_attributes

Lists the currently-defined attributes.

SYNTAX

```
status list_attributes
[-application]
[-class class_name]
[-nosplit]
```

Data Types

class_name string

ARGUMENTS

-application

Lists application attributes and user-defined attributes.

-class *class_name*

Limits the listing to attributes of a single class. Valid classes are **all**, **bound**, **cell**, **clock**, **design**, **ilms**, **lib**, **lib_cell**, **lib_pin**, **net**, **phys_cluster**, **phys_design**, **phys_layer**, **physcell**, **physnet**, **physport**, **pin**, **placement_bound**, **port**, **power_domain**, **power_switch**, **region**, **supply_net**, and **supply_port**.

-nosplit

Prevents lines from being split when column fields overflow. Most of the attribute information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line in the correct column.

DESCRIPTION

The **list_attributes** command displays an alphabetically-sorted list of currently-defined attributes. The attributes are divided into two categories: application-defined and user-defined. By default, **list_attributes** lists all user-defined attributes.

Using the **-application** option adds all application attributes to the listing. Since there are many application attributes, you can limit the listing to a specific object class by using the **-class** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example listing of some attributes defined with the **define_user_attribute** command:

```
prompt> list_attributes
*****
Report : attribute definition
Design :
*****

Attributes:
  r - read-only
  u - user-defined

Attribute name      Class  Type  Attributes
-----
DDB_EXTRACT_HIGH_FAN_OUT_THRES_ATTR lib_cell integer u
DDB_HIER_TOTAL_UNGROUP_AREA_ATTR   lib_cell integer u
DDB_REAL_METALFILL_EXTR_ATTR       lib_cell integer u
DDB_VIR_SHIELD_EXTR_ATTR           lib_cell boolean u
...
worst_slack                port  float  u
write_port_name            cell   string u
write_sched_step           cell   integer u
-----
1
```

The following example adds application-defined attributes, but limits the listing to net attributes only:

```
prompt> list_attributes -application -class net
*****
Report : attribute definition
        -application
Design :
*****

Attributes:
  a - application-defined
  r - read-only
  u - user-defined

Attribute name      Class  Type  Attributes
-----
actual_fall_transition_max    net    float  a
actual_fall_transition_min    net    float  a
actual_max_net_capacitance    net    float  a
actual_min_net_capacitance    net    float  a
actual_rise_transition_max    net    float  a
actual_rise_transition_min    net    float  a
...
worst_length_attr           net    float  a
x_length                    net    integer a
```

```
y_length          net    integer  a
-----
1
```

SEE ALSO

define_user_attribute(2)
get_attribute(2)
remove_attribute(2)
report_attribute(2)
set_attribute(2)

list_commands

Displays quick help for one or more commands.

SYNTAX

```
string list_commands  
  [-verbose]  
  [-groups]  
  [-bg]  
  [pattern]
```

Data Types

pattern string

ARGUMENTS

-verbose

Displays the command options; for example *command_name* **-help**.

-groups

Displays a list of command groups only.

-bg

Displays the list of commands supported within redirect -bg.

pattern

Displays commands matching the specified pattern.

DESCRIPTION

The **list_commands** command is used to get quick help for one or more commands or procedures. This is not the same as the **man** command that displays reference manual pages for a command. There are many levels of help.

By typing the **list_commands** command, a brief informational message is printed followed by the available command groups.

List all of the commands in a group by typing the group name as the argument to **list_commands**. Each command is followed by a one-line description of the command.

To get a one-line help description for a single command, type **list_commands** followed by the command name. You can specify a wildcard pattern for the name; for example, all commands containing the string "alias". Use the **-verbose** option to get syntax help for one or more commands. Use the **-groups** option to show only the command groups in the application. This option cannot be combined with any other option.

EXAMPLES

The following example lists the commands by command group:

```
prompt> list_commands
```

Specify a command name or wild card pattern to get help on individual commands. Use the '-verbose' option to get detailed option help for a command. Commands also provide help when the '-help' option is passed to the command.'

You can also specify a command group to get the commands available in that group. Available groups are:

```
Procedures:      Miscellaneous procedures
Help:             Help commands
Builtins:         Generic Tcl commands
...
```

The following example displays the list of procedures in the Procedures group:

```
prompt> list_commands procedures
```

```
ls          # List files
sh          # Execute a shell command
```

The following example uses a wildcard character to display a one-line description of all commands beginning with a:

```
prompt> list_commands a*
```

```
alias       # Create a command which expands to words.
append      # Builtin
array       # Builtin
```

This example displays option information for the **source** command:

```
prompt> list_commands -verbose source
```

```
source      # Read a file and execute it as a script
[-echo]     (Echo all commands)
[-verbose]  (Display intermediate results)
file_name   (Script file to read)
```

SEE ALSO

man(2)
sh_help_shows_group_overview(3)

list_designs

Lists the designs available in memory.

SYNTAX

```
status list_designs  
  [design_list]  
  [-show_file]
```

Data Types

design_list list

ARGUMENTS

design_list

Specifies the designs to list. If this option is not specified, all designs are listed. When specifying designs, the wildcard characters * (asterisk) and ? (question mark) are supported. For details about using and escaping wildcard characters, see the **wildcards** man page.

-show_file

Specifies that designs are shown according to the file in which they reside. The **-show_file** option is typically used when 2 or more designs share the same name.

DESCRIPTION

The **list_designs** command displays the designs currently read in dc_shell. In the design listing, the current design is designated by an asterisk (*).

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **list_designs**:

list_designs

```
prompt> list_designs
ADDER      FULL_SUBTRACTOR  HALF_SUBTRACTOR  TOP
FULL_ADDER  HALF_ADDER      SUBTRACTOR
```

```
prompt> current_design TOP
Current design is 'TOP'.
```

The following example shows that the current design is denoted with an asterisk:

```
prompt> list_designs
ADDER      FULL_SUBTRACTOR  HALF_SUBTRACTOR  TOP (*)
FULL_ADDER  HALF_ADDER      SUBTRACTOR
```

The following example lists all designs that end with `_ADDER`:

```
prompt> list_designs "**_ADDER"
HALF_ADDER  FULL_ADDER
```

The following example uses the `-show_file` option to differentiate between designs having the same name:

```
prompt> list_designs
ADDER      FULL_SUBTRACTOR  HALF_SUBTRACTOR  TOP
FULL_ADDER  HALF_ADDER      SUBTRACTOR
```

```
prompt> copy_design TOP example.db:TOP
Copying design 'TOP' to 'example.db:TOP'
```

```
prompt> list_designs
ADDER      FULL_SUBTRACTOR  HALF_SUBTRACTOR  TOP
FULL_ADDER  HALF_ADDER      SUBTRACTOR      TOP (*)
```

```
prompt> list_designs -show_file
/usr/users/bill/test/designs/top.db
ADDER      FULL_SUBTRACTOR  HALF_SUBTRACTOR  TOP (*)
FULL_ADDER  HALF_ADDER      SUBTRACTOR
```

```
example.db
TOP
```

SEE ALSO

current_design(2)
list_instances(2)
wildcards(3)

list_dont_touch_types

Lists the currently defined dont_touch types.

SYNTAX

```
status list_dont_touch_types  
[-class class_name]
```

Data Types

class_name string

ARGUMENTS

-class *class_name*

Limits the list of dont_touch types to either the cell or net object class. The value for **-class** is a class name. The supported class names are **cell** and **net**.

If you do not use this option, the command lists the dont_touch types for both the cell and net object classes.

DESCRIPTION

The **list_dont_touch_types** command displays an alphabetically sorted list of currently defined cell and net dont_touch types and their descriptions.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following command generates the list of dont_touch types for both cells and nets.

```
prompt> list_dont_touch_types
```

```
*****
```

```
Report : Types of dont_touch
```

Version: E-2010.12-SP2

Date : Thu Mar 3 12:06:02 2011

Class	Type	Description

net	cts_synthesized	Net is synthesized with clock tree synthesis
net	dft_scanbuf	Net connected to input pin of scan compression buffer cell
net	dtn	Net in dont_touch network set by set_dont_touch_network command
net	dtn_charz	Net in dont_touch network through characterization
net	fp_abutted	Net is floorplan abutted net
net	fp_border	Net is dangling on floorplan border
net	idn	Net in ideal network set by set_ideal_network command
net	ilm	Net is part of an interface logic model
net	inh_parent	Net inherits dont_touch from parent
net	mv_ao	Net has derived dont_touch from always-on synthesis
...		
cell	cg_mo	Clock-gating cell has dont_touch derived from map_only setting
cell	charz	Cell has dont_touch derived from characterization
cell	dft_scan	Cell is a DFT scan cell
cell	dft_scandef	Cell is SCANDEF generation cell
cell	dft_scg	DFT scan cell has dont_touch because it is replaced by clock-gating cell
cell	dt_conn	Cell connected to dont_touch net
cell	dtn	Cell in dont_touch network set by set_dont_touch_network command
...		

1		

SEE ALSO

```
report_dont_touch(2)
get_dont_touch_nets(2)
get_dont_touch_cells(2)
set_dont_touch(2)
set_dont_touch_network(2)
```

list_duplicate_designs

Lists designs that have the same design name in dc_shell.

SYNTAX

status **list_duplicate_designs**

ARGUMENTS

The **list_duplicate_designs** command has no arguments.

DESCRIPTION

This command displays a list of designs currently in dc_shell that have the same name as other designs also currently in dc_shell.

The command returns 1 if multiple designs with the same name are detected. The command returns 0 if no designs in dc_shell memory share the same design name.

EXAMPLES

In the following example, **list_duplicate_designs** lists 2 designs that have the same name in dc_shell memory:

```
prompt> list_duplicate_designs
```

```
Warning: Multiple designs in memory with the same design name.
```

Design	File	Path
-----	----	----
seq2	A.db	/usr/joe/design
seq2	B.db	/usr/joe/design

1

In the following example, **list_duplicate_designs** reports that there are no designs with the same name in dc_shell memory:

```
prompt> list_duplicate_designs
```

```
No duplicate designs found.
```

```
0
```

SEE ALSO

`current_design(2)`
`get_designs(2)`
`list_designs(2)`

list_files

Lists the files that are loaded into memory.

SYNTAX

status **list_files**

ARGUMENTS

The **list_files** command has no arguments.

DESCRIPTION

The **list_files** command lists the files loaded in memory.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the output from the **list_files** command:

```
prompt> list_files
File          Path
----          -
adder.db      /remote/usr4/joeuser/designs
clock.db      /remote/usr4/joeuser/designs
gtech.db      /remote/usr4/joeuser/libraries
lsi_10k.db    /remote/usr4/joeuser/libraries
1
```

SEE ALSO

`list_designs(2)`

`list_licenses(2)`

list_hdl_blocks

Lists the HDL blocks available in memory.

SYNTAX

```
int list_hdl_blocks  
  [-nosplit]  
  [-all_objects]  
  [instance_list]
```

Data Types

instance_list list

ARGUMENTS

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report. Most of the information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line in the correct column.

-all_objects

Displays all objects in each HDL block when this option is specified. By default, only up to 10 objects in each HDL block is displayed.

instance_list

Specifies the instances to display HDL blocks. If this option is not specified, the HDL blocks in the top-level instance are listed. This option supports the wildcard characters "*" and "?". For information about using and escaping wildcard characters, see the **wildcards** man page.

DESCRIPTION

The **list_hdl_blocks** command displays the HDL blocks in the current design.

The **list_hdl_blocks** command is not affected by the **current_instance** command. To report HDL blocks in lower-level blocks, you must use the *instance_list* argument.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **list_hdl_blocks** command:

```
prompt> list_hdl_blocks
```

```
list_hdl_blocks: hdl_block_2
```

```
Group          Objects
```

```
-----  
block_47       NIL
```

```
block_47/z_reg z_reg[4] z_reg[5] z_reg[6] z_reg[7]
```

```
block_03       NIL
```

```
block_03/z_reg z_reg[0] z_reg[1] z_reg[2] z_reg[3]  
-----
```

SEE ALSO

wildcards(3)

list_instances

Lists the instances in the current design or current instance.

SYNTAX

```
status list_instances
  [instance_list]
  [-hierarchy]
  [-max_levels num_levels]
  [-full]
  [-config]
```

Data Types

<i>instance_list</i>	list
<i>num_levels</i>	integer

ARGUMENTS

instance_list

Specifies the instances to list. By default, all instances in the current design or current instance are listed.

-hierarchy

Lists all levels of instance hierarchy. By default, only the current level of hierarchy is listed.

-max_levels *num_levels*

Modifies the **-hierarchy** option by specifying a limit to the number of levels of hierarchy that are listed. You must use the **-hierarchy** option if you use the **-max_levels** option.

-full

Displays the full hierarchy. By default, if there is a submodule in multiple locations in a hierarchy, its components are listed only once with ellipses (...) indicating the contents of a previously displayed module.

-config

Reports the RTL design library, template name, and specializing parameters rather than the generated name of the design cell.

DESCRIPTION

The **list_instances** command lists the instances in memory. The command displays cells at the current level of hierarchy, as specified by the **current_design** and **current_instance** commands.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **list_instances** to display the available instances. The design that each instance references is shown in parentheses following the instance name.

```
prompt> current_design TOP
```

Current design is 'TOP'.

```
prompt> list_instances
```

U1 (ADDER) U2 (SUBTRACTOR)

```
prompt> current_instance U1
```

Current instance is 'TOP/U1'.

```
prompt> list_instances
```

U1 (FULL_ADDER) U2 (FULL_ADDER)

U3 (FULL_ADDER) U4 (FULL_ADDER)

```
prompt> list_instances {U1, U2}
```

U1 (FULL_ADDER) U2 (FULL_ADDER)

```
prompt> current_instance U2
```

"TOP/U1/U2"

```
prompt> list_instances ""
```

U1 (HALF_ADDER) U2 (HALF_ADDER) U3 (OR2)

The following examples use the **-hierarchy** option to display multiple levels of the instance hierarchy. All instances at each level of hierarchy are listed as a group.

```
prompt> current_instance
```

Current instance is the top level of design 'TOP'.

```
prompt> list_instances -hierarchy
```

U1 (ADDER)

 U1 (FULL_ADDER)

 U1 (HALF_ADDER)

 U1 (EO) U2 (AN2)

 U2 (HALF_ADDER)...

 U3 (OR2)

 U2 (FULL_ADDER)...

 U3 (FULL_ADDER)...

 U4 (FULL_ADDER)...

U2 (SUBTRACTOR)

 U1 (FULL_SUBTRACTOR)

 U1 (HALF_SUBTRACTOR)

```
U1 (EO)    U2 (IV)    U3 (AN2)
U2 (HALF_SUBTRACTOR) ...
U3 (OR2) ...
U2 (FULL_SUBTRACTOR) ...
U3 (FULL_SUBTRACTOR) ...
U4 (FULL_SUBTRACTOR) ...
```

prompt> **list_instances -hierarchy -max_levels 2**

```
U1 (ADDER)
U1 (FULL_ADDER) ...
U2 (FULL_ADDER) ...
U3 (FULL_ADDER) ...
U4 (FULL_ADDER) ...
U2 (SUBTRACTOR)
U1 (FULL_SUBTRACTOR) ...
U2 (FULL_SUBTRACTOR) ...
U3 (FULL_SUBTRACTOR) ...
U4 (FULL_SUBTRACTOR) ...
```

prompt> **current_instance U1/U1**

Current instance is 'TOP/U1/U1'.

prompt> **list_instances -hierarchy**

```
U1 (HALF_ADDER)
U1 (EO)    U2 (AN2)
U2 (HALF_ADDER) ...
U3 (OR2)
```

prompt> **list_instances -hierarchy -full**

```
U1 (HALF_ADDER)
U1 (EO)    U2 (AN2)
U2 (HALF_ADDER)
U1 (EO)    U2 (AN2)
U3 (OR2)
```

SEE ALSO

current_design(2)
current_instance(2)
list_designs(2)
wildcards(3)

list_libs

Lists the libraries available in memory.

SYNTAX

```
status list_libs
[lib_list]
```

Data Types

```
lib_list  list
```

ARGUMENTS

lib_list

Specifies the libraries to list. You can use the * (asterisk) and ? (question mark) wildcard characters when specifying the libraries to list. For information about using and escaping wildcards, see the **wildcards** man page. By default, the tool lists all libraries.

DESCRIPTION

This command displays all libraries that are currently available in memory.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example shows a simple use of the **list_libs** command. The library flagged with *M* is the maximum library used for maximum delay analysis. It corresponds to the library flagged with *m* used for minimum delay analysis.

```
prompt> set_min_library sc_max.db -min sc_min.db
```

```
prompt> set_min_library tt_max.db -min tt_min.db
```

```
prompt> list_libs
```

Logical Libraries:

Library	File	Path
cell	cells.db	/remote/usr4/joeuser/designs
gtech	gtech.db	/remote/usr4/joeuser/libraries
standard.sldb	standard.sldb	/remote/usr4/joeuser/libraries
M fs120_max	sc_max.db	/remote/usr4/joeuser/libraries
m fs120_min	sc_min.db	/remote/usr4/joeuser/libraries
M cb_max	tt_max.db	/remote/usr4/joeuser/libraries
m cb_min	tt_min.db	/remote/usr4/joeuser/libraries

SEE ALSO

- list_designs(2)
- list_instances(2)
- read_lib(2)
- report_lib(2)
- set_min_library(2)
- wildcards(3)

list_licenses

Displays a list of licenses currently checked out by the user.

SYNTAX

status **list_licenses**

ARGUMENTS

The **list_licenses** command has no arguments.

DESCRIPTION

The **list_licenses** command lists the licenses you currently have checked out. If more than one copy of a license key is checked out, the number of keys checked out is shown in parentheses following the license name.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the output from **list_licenses**:

```
prompt> list_licenses
```

```
Licenses in use:
```

```
DC-Expert (3)  
DC-Ultra-Features (3)  
DC-Ultra-Opt (3)  
Design-Compiler  
DesignWare
```

```
1
```

SEE ALSO

check_license(2)
get_license(2)
license_users(2)
remove_license(2)

list_size_only_types

Lists the currently defined size_only types.

SYNTAX

status **list_size_only_types**

DESCRIPTION

The **list_size_only_types** command displays an alphabetically sorted list of currently defined cell size_only types and their descriptions.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following command generates the list of size_only types.

```
prompt> list_size_only_types
```

```
*****
```

```
Report : Types of size_only
```

```
Version: H-2013.03
```

```
Date   : Mon Aug 20 14:39:42 2012
```

```
*****
```

Type	Description

abs	Block abstraction cell in sizing-only optimization mode
cg_mo	Clock-gating cell has size_only derived from map_only setting
cg_obs	Clock-gating observation cell
charz	Cell has size_only derived from characterization
cts_sg	Cell is a self-gating cell created by clock tree synthesis
dft_so	Cell has size_only because it has a test_dft_cell_is_size_only attribute
dtn_conn	Cell connected to dont_touch net
dtn_noprop_load	Load cell whose driver is dont_touch_network source with dont_touch_network_no_propagate attribute
dtn_src	Output pin of the cell is dont_touch_network source

dw_so	Cell with dw_size_only attribute
dw_struct	Cell in DesignWare structure to be preserved
footprint	Cell has size_only to only perform footprint-preserving sizing
fp_ft_buffer	Floorplan feed-through buffer cell
fp_group	Leaf cell connected to port in floorplan group
hdl_rp	Cell with hdl_rp_cell attribute
icg	Integrated clock gate cell
inh_parent	Cell inherits size_only from hierarchical parent cell
inh_ref	Cell inherits size_only from reference design or library cell
inplace	Cell in a placed cluster
other	Cell has size_only due to other internal reason
pdf_t_scan	PDEF scan cell has size_only
power_roi	Isolation operand cell
reg_bank	Register cell in user-created register bank
scan_chain_element	Cell is a scan chain element
scan_comp_pg	Power gating cell for scan compressor
scan_exact_reg	Scan register that needs an exact match
...	
upf_retention_net	Driver of retention save-net or restore-net
upf_supply_conn	Cell connected to a supply net has size_only to prevent the supply net from being removed
user	Cell has specific user-set size_only attribute

1

SEE ALSO

report_size_only(2)
 set_size_only(2)
 set_compile_directives(2)

list_test_models

Lists the designs in memory that have CTL test models attached to them.

SYNTAX

```
status list_test_models  
[-compressors]
```

ARGUMENTS

-compressors

Lists the compressors and decompressors that are contained in each CTL-modeled design.

DESCRIPTION

This command displays the designs in memory that have CTL test models attached to them. Designs that do not have CTL test model information are not listed. All designs in memory are analyzed, even if they are not a part of the current design or its hierarchy.

EXAMPLES

The following example lists the designs that have CTL test model information:

```
prompt> list_test_models  
des_unit          /remote/my_dir/des_unit_xtol/db/des_unit.db  
des_unit_SCCOMP_COMPRESSOR  /remote/my_dir/des_unit_xtol/  
                        des_unit_SCCOMP_COMPRESSOR.db  
des_unit_SCCOMP_DECOMPRESSOR /remote/my_dir/des_unit_xtol/  
                        des_unit_SCCOMP_DECOMPRESSOR.db
```

The following example lists the compressors and decompressors for each design:

```
prompt> list_test_models -compressors  
TOP              /path/my_dir/TOP.db  
Codecs:  
  
C_CORE1         /path/my_dir/TOP.db
```

Codecs:

C_CORE1_U_decompressor_ScanCompression_mode

C_CORE1_U_compressor_ScanCompression_mode

C_CORE2 /path/my_dir/TOP.db

Codecs:

C_CORE2_U_decompressor_ScanCompression_mode

C_CORE2_U_compressor_ScanCompression_mode

list_test_modes

Displays all of the test modes that are defined for the current design.

SYNTAX

status **list_test_modes**

ARGUMENTS

The **list_test_modes** command has no arguments.

DESCRIPTION

The **list_test_modes** command displays all of the modes defined for the current design. The **define_test_mode** command is used to define a mode of operation for a design.

If a design does not have any test modes defined, it displays the message "Design has no test modes."

EXAMPLES

The following is an example of the test modes report:

```
prompt> list_test_modes
```

```
Test Modes
```

```
=====
```

```
    Name: ScanCompression_mode
```

```
    Type: InternalTest
```

```
    Focus:
```

```
    Core des_unit_U_decompressor in decompress mode
```

```
    Core des_unit_U_compressor in compress mode
```

```
    Name: Internal_scan
```

```
    Type: InternalTest
```

```
    Focus:
```

Name: Mission_mode
Type: Normal

1

SEE ALSO

current_design(2)
current_test_mode(2)
define_test_mode(2)
insert_dft(2)

Iminus

Removes one or more named elements from a list and returns a new list.

SYNTAX

```
list Iminus
[-exact]
original_list
elements
```

Data Types

<i>original_list</i>	list
<i>elements</i>	list

ARGUMENTS

-exact

Specifies that the exact pattern is to be matched. By default, **Iminus** uses the default match mode of **Isearch**, the **-glob** mode.

original_list

Specifies the list to copy and modify.

elements

Specifies a list of elements to remove from *original_list*.

DESCRIPTION

The **Iminus** command removes elements from a list by using the element itself, rather than the index of the element in the list (as in **Ireplace**). The **Iminus** command uses the **Isearch** and **Ireplace** commands to find the elements and replace them with nothing.

If none of the elements are found, a copy of *original_list* is returned.

The **Iminus** command is often used in the translation of Design Compiler scripts that use the subtraction operator (-) to remove elements from a list.

EXAMPLES

The following example shows the use of the **lminus** command. Notice that no error message is issued if a specified element is not in the list.

```
prompt> set l1 {a b c}
a b c
```

```
prompt> set l2 [lminus $l1 {a b d}]
c
```

```
prompt> set l3 [lminus $l1 d]
a b c
```

The following example illustrates the use of **lminus** with the **-exact** option:

```
prompt> set l1 {a a[1] a* b[1] b c}
a a[1] a* b[1] b c
```

```
prompt> set l2 [lminus $l1 a*]
{b[1]} b c
```

```
prompt> set l3 [lminus -exact $l1 a*]
a {a[1]} {b[1]} b c
```

```
prompt> set l4 [lminus -exact $l1 {a[1] b[1]} ]
a a* b c
```

SEE ALSO

lreplace(2)
lsearch(2)

load_of

Returns the capacitance of the specified library cell pin.

SYNTAX

```
float load_of  
    library_cell_pin
```

Data Types

library_cell_pin string

ARGUMENTS

library_cell_pin

Specifies the name of the library cell pin whose capacitance is to be returned. The format for *library_cell_pin* is *library_name/lib_cell_name/pin_name*. The specified library must be already loaded into memory.

DESCRIPTION

This command returns the load of the specified library cell pin. It is primarily used as an argument to the **set_load** command. If the specified *library_cell_pin* cannot be found, a warning message is issued and 0.0 is returned.

EXAMPLES

The following command returns the load of the pin named A on the AND2 cell in the tech_lib library:

```
prompt> load_of tech_lib/AND2/A
```

The following example uses the **load_of** command with the **set_load** command. It sets the load on all output ports of the design to the value of the load on the A input pin of the INV cell in the tech_lib library:

```
prompt> set_load [load_of tech_lib/INV/A] [all_outputs]
```

SEE ALSO

`set_load(2)`

load_upf

Executes a script file containing UPF commands.

SYNTAX

```
status load_upf
  [gupf_file_name]
  [-scope instance_name]
  [-noecho]
  [-simulation_only]
  [-strict_check true | false]
  [-supplemental supf_file_name]
```

Data Types

<i>gupf_file_name</i>	string
<i>instance_name</i>	string
<i>supf_file_name</i>	string

ARGUMENTS

gupf_file_name

Specifies the name of the file containing UPF commands to be executed.

In UPF-prime (non-golden) mode, this argument is required.

In golden UPF mode, this argument is optional; it specifies the name of the golden UPF script file to execute, if any. To execute a supplemental UPF script file, use the **-supplemental** option.

-scope *instance_name*

Specifies the scope where the UPF commands are executed. By default, the commands are executed in the current scope.

In the golden UPF mode, this option specifies the scope where both the golden UPF file and supplemental UPF file are executed.

-noecho

Prevents the UPF commands from being displayed as they are executed. By default, the UPF commands are echoed on the screen as they are executed.

-simulation_only

Indicates that the commands in the UPF file are for simulation only. A tool that does not perform simulation reads the commands and preserves them for output with the **save_upf** command, but does not execute them.

-strict_check true | false

Specifies whether to perform strict name matching during execution of the UPF commands. Use this option only in the golden UPF mode.

Set this option to **true** when you load the golden UPF file into the synthesis tool for the very first time, when object names in the UPF commands are expected to match object names in the design exactly.

Set this option to **false** when you reapply the golden UPF file any time after the synthesis tool has operated on the design, when object names in the file are not expected to match object names in the design due to optimization, grouping, and wildcard name expansion. In this case, the tool identifies objects using rule-based matching and name map file, if any.

The default setting is **true**; if you do not use the **-strict_check** option in the golden UPF mode, the tool uses strict name checking.

-supplemental *supf_file_name*

Specifies the name of the supplemental UPF file to be read. This option is supported only in the golden UPF mode.

DESCRIPTION

The **load_upf** command executes the set of UPF commands in the specified UPF file. It works like the **source** command, but performs additional checking specifically for UPF power intent. In the golden UPF mode, using the **load_upf** command is the only way to execute UPF commands (other than query commands).

If you use the **-scope** option, the command first sets the scope to the specified instance, executes the UPF command script to apply the commands at that scope, and then restores the scope to its previous state.

The **load_upf** command executes the constraints as they are being read. If it encounters errors while the UPF file is being read, execution might stop, so that only some of the UPF constraints are applied.

The **load_upf** command supports the following UPF commands. Usage of some of these commands is restricted when reading the UPF file. In certain cases, some command options are not supported.

```

add_port_state
add_pst_state
bind_checker
connect_supply_net
create_upf2hdl_vct
create_hdl2upf_vct
create_power_domain
create_power_switch
create_pst
create_supply_net
create_supply_port
create_supply_set
load_upf
map_isolation_cell
map_level_shifter_cell
map_power_switch
map_retention_cell
name_format
save_upf
set_scope
set_isolation
set_isolation_control

```

```
set_retention
set_retention_control
set_domain_supply_net
set_design_top
set_level_shifter
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example loads the UPF script file named top.upf:

```
prompt> load_upf top.upf
```

```
Loading UPF file top.upf with version 1.0...
```

```
End loading UPF file top.upf
```

```
1
```

The following example shows the usage of the **-scope** option:

```
prompt> load_upf top.upf -scope A
```

```
Loading UPF file top.upf with version 1.0...
```

```
End loading UPF file top.upf
```

```
1
```

The following example reads the UPF script file named top2.upf, which contains commands that are not supported by UPF. CMD-005 error messages are generated for the unsupported commands.

```
prompt> load_upf top2.upf
```

```
Loading UPF file top2.upf with version 1.0...
```

```
link_design
```

```
Error: Unknown command `link_design` (CMD-005)
```

```
End loading UPF file top2.upf
```

```
0
```

The following example shows the behavior of the **load_upf** command when the UPF file contains commands not supported by the tool. One UPF message is generated for every unsupported command. When the execution of the **load_upf** command is complete, the tool also reports a summary of all unsupported commands encountered in the UPF file.

```
prompt> load_upf t2.upf
```

```
Loading UPF file t2.upf with version 1.0...
```

```
set_logic_zero portA
```

```
Warning: Constraint 'set_logic_zero' is not supported by dc_shell. (UPF-042)
```

```
set_logic_zero portB
```

```
Warning: Constraint 'set_logic_zero' is not supported by dc_shell. (UPF-042)
```

```
set_logic_one portC
```

Warning: Constraint 'set_logic_zero' is not supported by dc_shell. (UPF-042)

Summary of unsupported constraints:

Information: Ignored 1 unsupported 'set_logic_one' constraint. (UPF-043)

Information: Ignored 2 unsupported 'set_logic_zero' constraints. (UPF-043)

End loading UPF file t2.upf

1

The following example illustrates the use of the **-noecho** option:

```
prompt> load_upf 5.upf -noecho
```

Loading UPF file 5.upf with version 1.0...

End loading UPF file 5.upf

1

The following example illustrates how to load the golden UPF file for the first time, with **-strict_check** set to **true**:

```
prompt> load_upf g.upf -strict_check true
```

Loading Golden UPF file g.upf ...

End loading UPF file g.upf

1

The following example illustrates how to reapply the golden UPF file:

```
prompt> load_upf g.upf -strict_check false
```

Loading Golden UPF file g.upf ...

End loading UPF file g.upf

1

The following example shows how to load a supplemental UPF file in the golden UPF mode:

```
prompt> load_upf -supplemental s.upf
```

Loading Supplemental UPF file s.upf ...

End loading UPF file s.upf

1

SEE ALSO

save_upf(2)

set_scope(2)

enable_golden_upf(3)

ls

Lists the contents of a directory.

SYNTAX

string **ls**

ARGUMENTS

The **ls** command has no arguments.

DESCRIPTION

If no argument is specified, the contents of the current directory are listed. For each *filename* matching a directory, **ls** lists the contents of that directory. If *filename* matches a file name, the file name is listed.

EXAMPLES

```
prompt> ls *.db *.pt
```

```
test1.pt    c1.db      c3.db      c5.db
```

```
test2.pt    c2.db      c4.db      c6.db
```

SEE ALSO

cd(2)

magnet_placement

Performs magnet placement.

SYNTAX

```
status magnet_placement
  [-move_fixed]
  [-mark_fixed]
  [-exclude_cells object_list]
  [-avoid_soft_blockages]
  [-stop_by_sequential_cells]
  [-exclude_buffers]
  [-logical_level level]
  [-stop_points object_list]
  [-cells object_list]
  [-align]
  magnet_objects
```

Data Types

```
object_list    collection
level          integer
magnet_objects collection
```

ARGUMENTS

-move_fixed

Specifies that a cell marked fixed can be moved. A warning occurs when a fixed cell is being moved. By default, fixed cells cannot be moved.

-mark_fixed

Specifies that the cells are to be marked fixed after placement. By default, no cells are marked fixed.

-exclude_cells *object_list*

Instructs the command to skip the specified cells when pulling cells toward the magnet object.

-avoid_soft_blockages

Specifies that cells should not be placed over soft blockages. By default, soft blockages are ignored.

-stop_by_sequential_cells

Specifies to stop pulling cells closer to the magnet objects when a sequential cell is encountered during level traversal. This

option should be used together with **-logical_level**

-exclude_buffers

Specifies not to consider buffers and inverters when calculating the level. They are pulled like normal cells but are skipped over for level determination. By default, buffers and inverters are considered as one level of logic. This option should only be used together with **-logical_level**.

-logical_level *level*

Specifies the number of logical levels to pull. By default, *level* is 1 (only the first-level cells are pulled).

This option is mutually exclusive with **-stop_points**.

-stop_points *object_list*

Specifies to pull only the cells that are on the timing paths between the magnet objects and the stop-points objects. The objects can be lists of any pin, port, or cell objects.

This option is mutually exclusive with **-logical_level** and **-exclude_buffers**.

-cells *object_list*

Specifies to pull only the cells in the specified object list. You specify only cell objects in the object list.

This option is mutually exclusive with **-logical_level**, **-exclude_buffers**, **-stop_by_sequential_cells**, and **-stop_points**.

-align

Specifies to place the cells in a horizontal or vertical alignment with respect to the pin of the macro.

magnet_objects

Specifies the objects to become magnets. The objects can be a fixed macro cell, a fixed standard cell, a pin of a fixed macro cell, or an I/O pin. Objects like vias and blockages are not allowed to be magnets.

DESCRIPTION

The **magnet_placement** command allows you to define a set of objects as magnets and pull the neighboring cells up to a specified logical level closer to the magnet. Specify a large macro with many placeable standard cell neighbors. Magnet placement can be used on this type of macro to ensure that the standard cells are placed close to the macro, resulting in better timing and congestion.

If there are a large number of high fanout nets, the **magnet_placement** command might pull a large number of cells resulting in long runtime. To avoid this situation, set the **magnet_placement_fanout_limit** variable to a reasonable value. The default is 1000.

The effect of magnet placement is not preserved from one invocation to another; for example, if cells A and B are made magnets and there are cells that are connected to both of them through possibly multiple levels. Running **magnet_placement A** and then **magnet_placement B** could result in a different result from running **magnet_placement [get_cells "A B"]** together. Doing the two cells together gives the correct result.

By default, only the logic before sequential elements is pulled by magnet placement.

If the **magnet_placement** command finds a fixed cell and the **-move_fixed** option is not specified, the fixed cell and all cells behind it are not pulled.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example performs magnet placement on the *INST_1* cell:

```
prompt> magnet_placement [get_cells "INST_1"]
```

The following example performs magnet placement and marks the moved cells as dont_touch placement:

```
prompt> magnet_placement -mark_fixed [get_cells "INST_2"]
```

The following example pulls two levels of cells to *INST_2*. Buffers (if any) are pulled but skipped over when calculating levels.

```
prompt> magnet_placement -exclude_buffers -logical_level 2 \  
[get_cells "INST_2"]
```

The following example works in default hierarchy mode:

```
prompt> magnet_placement INST1 -cells [get_cells module*] \  
-hierarchy_mode default
```

SEE ALSO

set_cell_location(2)

man

Displays reference manual pages.

SYNTAX

string **man**
topic

Data Types

topic string

ARGUMENTS

topic

Specifies the subject to display. Available topics include commands, variables, and error messages

DESCRIPTION

The **man** command displays the online manual page for a command, variable, or error message. You can write man pages for your own Tcl procedures and access them with the **man** command by setting the **sh_user_man_path** variable to an appropriate value. See the man page for the **sh_user_man_path** variable for details.

EXAMPLES

The following command displays the man page for the **echo** command:

```
prompt> man echo
```

The following command displays the man page for the error message CMD-025:

```
prompt> man CMD-025
```

SEE ALSO

[help\(2\)](#)

[sh_user_man_path\(3\)](#)

map_isolation_cell

Specifies how to map or remap the isolation and enable level-shifter cells belonging to the specified isolation strategy.

SYNTAX

```
status map_isolation_cell  
  isolation_strategy  
  -domain power_domain  
  -lib_cells lib_cells
```

Data Types

<i>isolation_strategy</i>	string
<i>power_domain</i>	string
<i>lib_cells</i>	list

ARGUMENTS

isolation_strategy

Specifies the UPF isolation strategy name. The name of the isolation strategy must be unique within the specified power domain.

-domain *power_domain*

Specifies the name of the power domain name to which this isolation strategy belongs.

-lib_cells *lib_cells*

Specifies a list of the target library cells to be used for the isolation mapping.

DESCRIPTION

This command defines how the **compile** command performs the mapping of an isolation cell. All isolation cells that match the strategy are mapped or remapped to one of the cells specified with the **-lib_cells** option. Resizing is done if required, but the choice is limited only to the library cells specified with the **-lib_cells** option.

You can also specify enable level-shifter cells using the **-lib_cell** option. If the isolation cells are later changed to enable level-shifter cells in the technology library, the cells can only be mapped to the one of the enable level-shifter cells specified with this command.

If you specify cells that invert the data pin, they will only be considered for remapping pre-instantiated isolation cells that also invert the data pin. Pre-instantiated isolation cells that invert the data pin are only remapped to isolation cells that also invert the data pin.

EXAMPLES

In the following example, the isolation cells belonging to the strategy named *isolation_1* can only be mapped to the *LIB_HICLAMP0X2* or *LIB_HICLAMP0X4* library cells:

```
prompt> map_isolation_cell isolation_1 -domain PD1 \  
        -lib_cells {LIB_HICLAMP0X2, LIB_HICLAMP0X4}
```

SEE ALSO

set_isolation(2)
set_isolation_control(2)

map_level_shifter_cell

Specifies that the level-shifter cells belonging to the specified strategy can only be mapped to a subset of the library cells.

SYNTAX

```
status map_level_shifter_cell  
  level_shifter_strategy  
  -domain power_domain  
  -lib_cells lib_cells
```

Data Types

<i>level_shifter_strategy</i>	string
<i>power_domain</i>	string
<i>lib_cells</i>	list

ARGUMENTS

level_shifter_strategy

Specifies the UPF level-shifter strategy name. The level-shifter strategy name must be unique within the specified power domain.

-domain *power_domain*

Specifies the power domain name to which this level-shifter strategy belongs.

-lib_cells *lib_cells*

Specifies a list of library cells that can be used to map to the level-shifter cells.

DESCRIPTION

This command defines how the **compile** command maps the level-shifter cells. All level-shifter cells belonging to the specified strategy are mapped to one of the library cells specified in the **-lib_cells** list. This subset is honored during resizing. If a valid solution cannot be reached using only the specified library cells, no level-shifter cells are inserted. For example, if the level shifters specified in the **-lib_cells** option cannot be used to shift between the required voltage levels, the tool does not insert any level-shifter cell.

For mapping the enable level-shifter cells, use the **map_isolation_cell** command.

EXAMPLES

In the following example the level-shifter cells belonging to the strategy named *VL_1* can only be mapped to the *LIB_LSHIX2* or *LIB_LSHIX4* library cells:

```
prompt> map_level_shifter_cell LVL_1 -domain PD1 \  
-lib_cells {LIB_LSHIX2, LIB_LSHIX4}
```

SEE ALSO

map_isolation_cell(2)
set_level_shifter(2)

map_power_switch

Defines which power switch library cells to use for the mapping of the given UPF power switch.

SYNTAX

```
status map_power_switch  
  switch_name  
  -domain domain_name  
  -lib_cells list
```

Data Types

```
switch_name  string  
domain_name  string  
list         list
```

ARGUMENTS

switch_name

Specifies the name of the power switch to be mapped using the specified library cells. The specified name must be a simple name and follow the name specified in the **create_power_switch**. This option is required.

-domain *domain_name*

Specifies the power domain where this power switch was created. This option is required.

-lib_cells *list*

Specifies a list of target library cells to use for the power switch cell mapping. This option is required.

DESCRIPTION

This command is used to explicitly specify which library power switch cells are mapped for the corresponding UPF power switch.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example maps the *switch_1A* power switch in the power domain named *myPowerDomain* with the *lib2A/switch2A* library cell:

```
prompt> map_power_switch switch_1A \  
        -domain myPowerDomain -lib_cells lib2A/switch2A  
1
```

SEE ALSO

`create_power_switch(2)`

map_retention_cell

Defines how to map the unmapped sequential cells to retention cells for the specified UPF retention strategy of the power domain.

SYNTAX

```
status map_retention_cell  
  retention_strategy  
  -domain power_domain  
  [-lib_cells lib_cells]  
  [-lib_cell_type lib_cell_type]  
  [-elements objects]  
  [-lib_model_name name {-port port_name net_ref*}]
```

Data Types

<i>retention_strategy</i>	string
<i>power_domain</i>	string
<i>lib_cells</i>	list
<i>lib_cell_type</i>	string
<i>objects</i>	list
<i>name</i>	string
<i>port_name</i>	string
<i>net_ref</i>	string

ARGUMENTS

retention_strategy

Specifies the UPF retention strategy name. The retention strategy name must be unique within the specified power domain.

-domain *power_domain*

Specifies the power domain name to which this UPF retention strategy is applied.

-lib_cells *lib_cells*

Specifies a list of target library *lib_cells* to be used for the retention mapping.

-lib_cell_type *lib_cell_type*

Specifies the retention type to be used for the retention mapping.

-elements *objects*

Specifies that retention mapping only applies to the sequential cells within the specified objects. The objects can be a list of hierarchical cells, leaf cells, HDL blocks, or register signals of a Verilog always block.

-lib_model_name *name* {-port *port_name* *net_ref*}*

Specifies the retention library cell or behavioral model and associated port connectivity for verification tools.

DESCRIPTION

This command specifies how the **compile** command performs retention mapping. If **-elements** is not specified, then the mapping applies to all sequential cells to which the specified retention strategy applies.

The retention cell specified using the **-lib_model_name** option is used only by the verification tools. It is not used by the implementation tools. The **-lib_model_name** option is not written out in the UPF' file.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example maps the sequential cells under the retention_1 retention strategy to the CLK_LOW retention type:

```
prompt> map_retention_cell retention_1 \
        -domain PD1 -lib_cell_type CLK_LOW
```

The following example maps the sequential cells under the retention_2 retention strategy to the RSDFCD1HVT and RSDFCD2HVT retention library cells. The **-lib_model_name** option specifies the retention library cell, port name, and net reference used by the verification tools.

```
prompt> map_retention_cell retention_2 \
        -domain PDM -lib_cells { RSDFCD1HVT RSDFCD2HVT } \
        -lib_model_name RSDFCDHVT { -port CP UPF_GENERIC_CLOCK \
            -port D ~UPF_GENERIC_DATA \
            -port VDD sn1 \
            -port TVDD retention_ref.power }
```

SEE ALSO

set_retention(2)
set_retention_control(2)

mem

Reports memory usage information.

SYNTAX

```
status mem  
  [-all]  
  [-verbose]
```

ARGUMENTS

-all

Reports the maximum of the peak memory usage among the main process and its child processes during the compile phase.

-verbose

Reports in detail the peak memory usage of the main process and each child process, including placement, extraction, and routing during the compile phase.

DESCRIPTION

This command reports the peak memory usage of processes. By default, it reports the peak memory usage of the main process in kilobytes. If you specify the **-all** option, it reports the maximum of the peak memory usage among the main process and its child processes for compile. If you specify both the **-all** and **-verbose** options, it also prints out the peak memory usage in megabytes for the main process and each child process.

Memory peak is defined as the memory high-water mark of processes. When the tool reports a memory peak, it means the maximum memory allocated from the operating system. When child processes exist, the tool reports the maximum number among the main process and its child processes.

Note that the **mem** command reports the peak memory usage, not the swap space usage of the process. It actually reports how much memory was allocated from the operating system at the peak point. Some of the memory can be swapped out by the operating system based on the system status.

IMPORTANT: Peak memory use of child processes is only tracked for the "compile" and "optimize_netlist" commands.

EXAMPLES

The following example shows the default command output:

```
prompt> mem  
102400
```

The following example shows the output when using the **-all** and **-verbose** options:

```
prompt> mem -all  
Multicore compile mem-peak: 110 Mb  
102400
```

```
prompt> mem -all -verbose  
Multicore compile mem-peak: 110 Mb  
Main process mem-peak: 100 Mb  
Child "placement" called 3 times, mem-peak: 60 Mb  
Child "extraction" called 3 times, mem-peak: 50 Mb  
102400
```

SEE ALSO

`cputime(2)`
`PSYN-508(n)`

merge_saif

Reads a list of SAIF files with their corresponding weights, computes the merged toggle rate and static probability, and annotates the switching activity for the nets, pins, and ports in the current design. The command then generates a merged output SAIF file.

SYNTAX

```
status merge_saif
  -input_list saif_file_and_weight_list
  [-instance_name inst_name]
  [-output merged_saif_name]
  [-simple_merge]
  [-ignore ignore_name]
  [-ignore_absolute ig_absolute_name]
  [-exclude exclude_file_name]
  [-exclude_absolute ex_absolute_file_name]
  [-unit_base unit_value]
  [-scale scale_value]
  [-khrate khrate_value]
  [-map_names]
```

Data Types

```
saif_file_and_weight_list  list
inst_name                  string
merged_saif_name          string
ignore_name               string
ig_absolute_name          string
exclude_file_name         string
ex_absolute_file_name     string
unit_value                string
scale_value               integer
khrate_value              float
```

ARGUMENTS

-input_list *saif_file_and_weight_list*

Specifies the name and the corresponding weight of the Switching Activity Interchange Format (SAIF) file list. The *saif_file_and_weight_list* argument is a list of items in the following format:

```
{-input name.saif -weight number}
```

The **-input** and **-weight** are keywords, use the **-input** option to specify the .saif file and use the **-weight** option to specify a number between 0 and 100. For example, in the following statement, the gate_back_1.saif and gate_back_2.saif files are weighted by

20% and 80%, respectively and the sum of all weights is equal to **100**.

```
prompt> {-input gate_back_1.saif -weight 20 \
        -input gate_back_2.saif -weight 80}
```

-instance_name *inst_name*

Specifies the instance name as it appears in each SAIF file of the current design instance to annotate. The command annotates the instance itself and all subinstances in the hierarchy of the specified instance.

-output *merged_saif_name*

Specifies the name of the output file. The output file is the merged SAIF file.

-simple_merge

Specifies that all nets, ports, and pins of the current design must be annotated by the specified SAIF file. It also specifies that a simple SAIF file data merge must be performed. You can use the **report_saif** command to verify that each SAIF file annotates 100% of the current design.

-ignore *ignore_name*

Specifies the name of an instance for which the switching activity should be ignored. Use this option to ignore switching activity within the hierarchy of that instance in each SAIF file. The ignored names are recognized by the **-instance_name** option.

-ignore_absolute *ig_absolute_name*

Specifies the name of an instance for which the switching activity should be ignored, but is not affected by the use of the **-instance_name** option. The command determines whether a net name is within the ignored hierarchy before applying the **-instance_name** option.

-exclude *exclude_file_name*

Specifies the name of a file that contains a list of names that should be ignored. Use this option to ignore switching activity for instances specified by the *exclude_file_name* argument. In the file specified with the *exclude_file_name* argument, each ignored name should be on a separate line. Do not mention the **-exclude** option in the file. The ignored names are recognized after the **-instance_name** option is applied.

-exclude_absolute *ex_absolute_file_name*

Specifies the name of a file that contains a list of absolute names to be ignored but which are not affected by the use of the **-instance_name** option.

-unit_base *unit_value*

Specifies the base synthesis timing unit for all SAIF files. The synthesis timing unit is obtained from the value specified with the **-scale** option. This value is used to scale the base unit. The valid values for *unit_value* are **ns**, **us**, or **ps**. The default is **ns**.

-scale *scale_value*

Specifies the value of the scaling factor for the base synthesis timing unit for all SAIF files. The synthesis timing unit is obtained by using *scale_value* to scale the base unit. The valid values for *scale_value* are **1**, **10**, or **100**. The default is **1**.

-khrate *khrate_value*

Specifies a default derating factor value to be used for internal problems for all SAIF files. If you do not use the **-khrate** option, the command uses a default derating factor of **0.5**.

-map_names

Specifies that the SAIF name mapping mechanism should be used to match objects in the SAIF file, with design objects. Since **saif_map** command is not supported in **icc_shell**, this option is not valid in **icc_shell**.

OBSOLETE ARGUMENTS

`-rtl_direct`

The flows involving forward RTL SAIF files, and flows involving the `\Bmerge_saif -rtl_direct` is obsolete.

`-strip_module annotated_instance_name`

Specifies the instance name to be annotated in the current design. This option is obsolete. Use the **-instance** option instead.

DESCRIPTION

The **merge_saif** command reads a list of SAIF files with their corresponding weights, computes the merged toggle rate and static probability, and annotates the switching activity, **toggle_rate**, and **static_probability** attributes for the nets, ports, and pins of the current design. The **merge_saif** command also generates a merged output SAIF file.

To identify the instance (and corresponding subinstances) to annotate, use the **-instance_name** option. To set the synthesis timing unit, use the **-unit_base** and **-scale** options, unless you are certain that the synthesis timing unit is 1ns. To specify a derating factor use the **-khrate** option. If you do not specify a derating factor, the tool uses a default value of 0.5.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example reads and merges weighted SAIF files, annotates switching activity for the current design, and generates an output SAIF file:

```
prompt> merge_saif -input_list \  
        {-input gate_back1.saif -weight 10 \  
        -input gate_back2.saif -weight 20 \  
        -input gate_back3.saif -weight 30 \  
        -input gate_back4.saif -weight 40} \  
        -instance E/UUT -output merged_saif.saif
```

SEE ALSO

`read_saif(2)`
`report_saif(2)`
`saif_map(2)`

mw_cel_collection

Describes the methodology for using mw_cel collection.

DESCRIPTION

How to specify mw_cel by name

A unique mw_cel can be specified by the following:

A[B[:C]]

Where:

A is the base name of the mw_cel, and the maximum length of A is 1023.

B is the view name of the mw_cel, and the maximum length of B is 31.

C is the version number. The version number starts at 1.

The following characters . (period) and ; (semicolon) are invalid in the base name, view name, or version of an mw_cel name.

The version of an mw_cel name is usually omitted. If the version is not specified, the latest version is assumed unless an individual command has a special interpretation. The latest version is the version with highest version number.

Because ; is a keyword in Tcl language, if the version is to be specified for an mw_cel, you must use proper quoting, such as double quotation marks, for example:

"top.CEL;1".

If you do not use double quotation marks, the command might appear to succeed, but might produce unexpected results. For example, in the following command:

```
prompt> open_mw_cel test1.CEL;1
Info: Opened "test1.CEL;2" from "/root/data/design" library
[1]
```

1 is considered as a separate command, and the **open_mw_cel** command actually opens the latest version, not version 1.

You can omit the view name. When you do this, you must also omit the version. If you do not specify the view name, the command will assume the CEL view and the latest version unless the individual command has a special interpretation.

Currently supported views

Currently, 5 kind of views are supported: CEL, FRAM, err, FILL and ILM. You can not specify the mw_cel of other views through the command line. However, some commands may have the ability to operate on other views internally.

Display of the mw_cel collection

The display of an mw_cel collection does not contain view and version. For example:

```
prompt> open_mw_cel test1
```

Info: Opened "test1.CEL;2" from "/root/data/design" library
{test1}

To get the view and version information, use the attribute version and view_name.

```
prompt> get_attribute [current_mw_cel] view_name  
CEL
```

```
prompt> get_attribute [current_mw_cel] version  
2
```

SEE ALSO

- close_mw_cel(2)
- collections(2)
- copy_mw_cel(2)
- create_mw_cel(2)
- current_mw_cel(2)
- get_mw_cels(2)
- open_mw_cel(2)
- remove_mw_cel(2)
- rename_mw_cel(2)
- save_mw_cel(2)

name_format

Specifies the prefix and suffix used for naming isolation cells and level shifters created by the tool.

SYNTAX

```
status name_format
[-isolation_prefix name]
[-isolation_suffix name]
[-level_shift_prefix name]
[-level_shift_suffix name]
```

Data Types

name string

ARGUMENTS

-isolation_prefix

Specifies the prefix to use for naming isolation cells. The default is "".

-isolation_suffix

Specifies the suffix to use for naming isolation cells. The default is "_UPF_ISO".

-level_shift_prefix

Specifies the prefix to use for naming level-shifter cells. The default is "".

-level_shift_suffix

Specifies the suffix to use for naming level-shifter cells. The default is "_UPF_LS".

DESCRIPTION

This command specifies the suffix and prefix used for naming new isolation cells and level shifters created by the tool. The name of the element that is being isolated or level-shifted is used as the base for the name. The prefix is added to the front and the suffix is added to the back to create the new name.

For example, an isolation cell that isolates a port named *p1* is named *snps_PDName__ISOName_snps_p1__UPF_ISO* by default, where "PDName", "ISOName", and "_UPF_ISO" represent the power domain name, isolation strategy name, and default suffix, respectively.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the prefixes for naming new isolation cells and level shifters:

```
prompt> name_format -isolation_prefix "MY_ISO" -level_shift_prefix "MY_LS"
```

SEE ALSO

set_isolation(2)
set_level_shifter(2)

open_mw_lib

Opens a Milkyway library.

SYNTAX

```
collection open_mw_lib  
  [-readonly | -write_ref]  
  mw_lib
```

Data Types

mw_lib string

ARGUMENTS

-readonly

Opens the Milkyway library only for reading. You cannot modify and save the library in this mode.

The **-readonly** and **-write_ref** options are mutually exclusive. The default is to open the main library as read/write and all reference libraries as read only.

-write_ref

Opens the reference library for writing. This option implies that the main library is opened for writing.

The **-readonly** and **-write_ref** options are mutually exclusive. The default is to open the main library as read/write and all reference libraries as read only.

mw_lib

Opens the Milkyway library.

DESCRIPTION

This command opens a Milkyway library. The two access permissions for an opened library are read only or read/write. Specifying the write permission implies that read permission is granted. If you specify the **-readonly** option, the command opens the main library and all reference libraries as read only. If you specify the **-write_ref** option, it opens the main library and all reference libraries as read/write.

If you open the library as read only, you can read from the library but you cannot write to the library.

Specifically, opening the Milkyway CEL view to write in it is not permitted. If you open the library as read/write, you can modify it.

The opened Milkyway library is automatically set to be the current Milkyway library.

You cannot open more than one main library at the same time. To open another main library, you must first close the previous main library.

This command returns a collection of Milkyway libraries if it succeeds.

See the man pages for the **set_mw_lib_reference** and **set_mw_technology_file** commands for information on setting or changing reference libraries and technology files.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example opens a Milkyway library named *access05* for writing in the current session:

```
prompt> open_mw_lib access05
{access05}
```

SEE ALSO

close_mw_lib(2)
copy_mw_lib(2)
create_mw_lib(2)
current_mw_lib(2)
open_mw_lib(2)
rebuild_mw_lib(2)
rename_mw_lib(2)
report_mw_lib(2)
set_mw_lib_reference(2)
set_mw_technology_file(2)
write_mw_lib_files(2)

optimize_netlist

Performs a high-effort optimization on the current design for better quality of results (QoR).

SYNTAX

```
status optimize_netlist  
  [-area]  
  [-no_boundary_optimization]  
  [-no_seq_output_inversion]
```

ARGUMENTS

-area

Performs monotonic gate-to-gate optimization to improve area without degrading timing or leakage. To maximize area benefits, run the **optimize_netlist -area** command as the final area recovery step at the end of the synthesis flow. You can also use the command on legacy netlists.

-no_boundary_optimization

Disables hierarchical boundary optimization. This option is superseded by **set_boundary_optimization** settings.

-no_seq_output_inversion

Disables sequential output inversion. This option is superseded by **set_register_output_inversion** settings.

SEE ALSO

compile_ultra(2)

optimize_registers

Performs retiming of sequential cells (edge-triggered registers or level-sensitive latches) on a mapped gate-level netlist; determines the placement of sequential cells in a design to achieve a target clock period; and minimizes the number of sequential cells while maintaining that clock period.

SYNTAX

```
status optimize_registers
  [-minimum_period_only]
  [-no_compile]
  [-sync_transform multiclass | decompose | dont_retime]
  [-async_transform multiclass | decompose | dont_retime]
  [-check_design [-verbose]]
  [-print_critical_loop]
  [-clock clock_name [-edge rise | fall]]
  [-latch]
  [-justification_effort low | medium | high]
  [-only_attributed_designs]
  [-delay_threshold target_clock_period]
```

Data Types

```
clock_name      string
target_clock_period float
```

ARGUMENTS

-minimum_period_only

Indicates that only the minimum period step of the retiming algorithm (minimum clock-period retiming) and not the minimum area (sequential cell-count optimization) step is to be executed. By default, both the minimum period and minimum area optimization steps are executed. This option is useful if you want a fast turnaround when trying to optimize the design's timing. The runtime is reduced, but the area results will not be optimal.

After you are satisfied with the timing, you can attempt to reduce the area by running the retiming command without this option.

-no_compile

Indicates that the default incremental logic synthesis step, normally performed after computation of the optimal sequential cell locations, is to be omitted. If you specify this option, no design rule fixing is performed. Generic sequential cells may remain in the design if the **-no_compile** option is specified. Using the **-no_compile** option, you can choose a logic compilation script that is adapted to your design instead of relying on the default used internally by **optimize_registers**. It is important to perform a logic synthesis step after sequential cell retiming to obtain the best possible timing results.

-sync_transform multiclass | decompose | dont_retime

Specifies which transformation method is used for synchronous sequential cells in the design. An edge-triggered register is synchronous if none of its input pins change the outputs asynchronously. A level-sensitive latch is considered synchronous if none of its input pins can change the outputs during the clock phase where the latch is not transparent.

Specifying **multiclass** transformation indicates that the identifiable synchronous clear, set, and enable functionality is moved with the synchronous sequential cells if they are moved during retiming. Sequential cells are classified according to their set, clear, and enable connections. The class of the sequential cells at the fanin or fanout of a combinational cell determines whether retiming across this cell can be performed.

Specifying **decompose** transformation indicates that synchronous sequential cells the design are transformed into an instance of a D-flip-flop or D-latch and additional combinational logic to create the necessary synchronous functionality. Only the D-flip-flop/D-latch instance can be moved during retiming.

Specifying **dont_retime** indicates that synchronous sequential cells are not moved during retiming. Their mapping might still be changed to a different flip-flop from the technology library.

The **dont_touch** attributes and retiming transformation attributes set on individual sequential cells override the value set in this option. To set retiming transform attributes, use the **set_transform_for_retiming** command.

The default argument for this option is **multiclass**.

-async_transform multiclass | decompose | dont_retime

Specifies which transformation method is used for asynchronous sequential cells in the design. An edge-triggered register is asynchronous if at least one of its input pins changes the outputs asynchronously. A level-sensitive latch is asynchronous if at least one of its inputs can change the outputs during the clock phase where the latch is not transparent.

Specifying the **multiclass** transformation indicates that the identifiable asynchronous clear and set functionality, as well as any synchronous set, clear, and enable functionality, is moved with the asynchronous sequential cells, if they are moved during retiming. Sequential cells are classified according to their set, clear, and enable connections. The class of the sequential cells at the fanin or fanout of a combinational cell determines whether retiming can be performed across this cell.

Specifying the **decompose** transformation indicates that asynchronous sequential cells in the design are transformed into an instance of a flip-flop or latch with asynchronous set and clear inputs, as necessary, and additional combinational logic to create the necessary synchronous functionality. Only the flip-flop/latch instances can be moved during retiming. They will be classified according to their synchronous set, clear, and enable functionality.

Specifying the **dont_retime** value indicates that asynchronous sequential cells are not moved during retiming. Their mapping might still be changed to a different flip-flop from the technology library.

The **dont_touch** attributes and retiming transformation attributes set on individual sequential cells override the value set in this option. To set retiming transform attributes, use the **set_transform_for_retiming** command.

The default value for this option is **multiclass**.

-check_design

Displays additional information about the design is before and after retiming. This information includes the number of cells in different categories (for example, hierarchy cells with **dont_touch** attributes or non-movable sequential cells) and more detailed information about the selection of the preferred flip-flop or latch. You use this information to help in troubleshooting if retiming does not show the expected results.

-verbose

Displays the explicit names of the cells along with the number of cells in each category for most categories of the **-check_design** option. This option can be used only with the **-check_design** option. The explicit naming of cells can help to locate a problem; however, the lists of output names might be long.

-print_critical_loop

Displays the critical loop of the design, as seen during retiming. The critical loop is defined as the sequence of directly-connected

combinational and sequential cells whose total combinational delay divided by the number of registers in the loop has a higher value than any other loop in the design. The critical loop limits the minimum clock period that can be achieved by retiming. Use this option to help with troubleshooting problem areas of the design if the intended clock period cannot be achieved with the given number of sequential cells in the design. If you are pipelining a data path, you might need to add pipeline stages in the HDL code.

-clock *clock_name*

Specifies the name of the clock whose sequential cells are to be retimed. The clock must not be a virtual clock, that is, it must have a clock port associated with it. The name of the clock is either the name specified in the **create_clock** command or the name of the clock port, if the **create_clock** command had no name specified. The registers of the clock are all sequential cells which are triggered by this clock. The connection from the clock port to the sequential cell can be through clock-gating cells, buffer cells, and inverter cells. If the **-clock** option is specified and edge-triggered registers are retimed, registers of other clocks are not retimed. If level-sensitive latches are retimed and the **-clock** is specified, the latches driven by this clock as well as those driven by other clocks needed to complete a two-phase clock system are retimed. If edge-triggered registers are retimed, only registers triggered by one specific edge of the clock are retimed. By default the registers triggered by the rising edge are retimed. A different edge can be specified using the **-edge** option.

-edge *rise* | *fall*

Specifies whether the registers triggered by the rising or the falling edge of the clock are to be retimed. This option can be used only with the **-clock** option. When level-sensitive latches are retimed, this option has no effect.

-latch

Specifies that level-sensitive latches are to be retimed instead of edge-triggered sequential cells (flip-flops). If this option is used, the edge-triggered sequential cells in the design are not moved. In order to be able to retime latches they must be driven by a symmetrical two-phase clocks system. Latches that are used to prevent glitches in gated clocks are not moved, even if the **-latch** option is used. These latches are in the fanin of clock-gating cells.

-justification_effort *low* | *medium* | *high*

Specifies the justification effort level to be used during backward justification of registers. You can specify either **low**, **medium** or **high** as a value for this option. Specifying **low** ensures that justification terminates very quickly, but the QoR may be bad. Specifying **medium** might give good QoR with a larger runtime. Specifying **high** might give the best QoR without any regard for runtime. The default value for this option is **medium**.

-only_attributed_designs

Specifies that instead of the top-level design, only instances of those designs in the hierarchy below the current designs that have the **optimize_registers** attribute set are retimed. The **optimize_registers** attribute can be set by using the **set_optimize_registers** command. The instances with attributes are retimed using the constraints derived from their environment.

-delay_threshold *target_clock_period*

Instructs *Design Compiler* to improve paths with delays less than worst delay as given in critical loop report. Design Compiler makes additional retiming passes until all such paths are optimized to operate at given target clock period.

DESCRIPTION

The **optimize_registers** command operates by default on the current design. If the **-only_attributed_designs** option is used, it will work only inside those hierarchical cells of the current design that are instances of designs having the **optimize_registers** attribute. The command operates in two phases. During the first phase, it performs retiming by moving the sequential cells in the design to meet a target clock period and minimize the number of sequential cells while maintaining that clock period. The second phase consists of an incremental compile, which adjusts the design to the changed fanout structure. By default, **optimize_registers** uses

the clock period of the clocks being retimed, if available. Otherwise, **optimize_registers** returns without moving registers. The final incremental compile targets the clock period defined by the **create_clock** command.

If the design has multiple clocks that are not virtual clocks and the **-clock** option has not been used, sequential cells for all clocks are retimed during the first phase of retiming. When retiming edge-triggered registers, the retiming is performed one clock at a time. When retiming level-sensitive latches the retiming is performed for sets of clocks which together with their latches form a symmetric two-phase clock system. Clocks with a larger clock period are retimed before clocks with a smaller clock period. If two clocks have the same clock period, the clock with the larger number of movable sequential cells is retimed first. When retiming edge-triggered registers for each single clock, the registers triggered by the rising edge are retimed before those triggered by the falling edge. Note that this default order may not yield the best possible results. Also, retiming all clocks in phase one means that there will be no incremental optimization of the combinational logic when retiming different clocks. Therefore, it is recommended that you determine the best order for retiming clocks and apply it using multiple runs of **optimize_registers** using the **-clock** option.

The **optimize_registers** command has the following requirements:

- A gated clock must be derived from one of the design's clock ports or another gated clock through a unate clock-gating cell (usually an logic AND or logic OR gate). The clock gating control logic may contain latches. The clock network between the base clock port and gated clocks may contain buffer and inverter cells.
- Flip-flops and master-slave elements cannot be present in the same design. Master and slave clock waveforms cannot overlap.
- All clock pins of all flip-flops in the design must be connected to their base clock or a gated clock derived from the base clock in the following way: The connection from the clock origin (that is, the clock port or the clock-gating cell) to the clock pins can contain buffer and inverter cells with one input and one output pin. All clock pins must be connected either by an even number (including zero) or odd number of inverters to the clock origin. Buffer and inverter cells on the clock network may be removed during retiming. Therefore, any existing clock tree must be resynthesized.
- All sequential cells must be single bit, or it must be possible to decompose them into single-bit registers.
- Only certain FPGA technologies are supported.
- Designs cannot contain a combinational loop.
- Set the timing constraints for the design in the following way: The external clock ports of the design must have a clock constraint created by the **create_clock** command. These are called the base clocks. All primary inputs of the design must have a non-negative input delay relative to one or more of the base clocks. All primary outputs of the design must have a non-negative output delay relative to one of the base clocks. Negative input and output delays are tolerated, but the quality of the final retiming result may be worse than expected. Point-to-point timing exceptions as created by the **set_false_path**, **set_multicycle_path**, **set_max_delay**, and **set_min_delay** commands are respected but their presence may reduce the quality of results. In the presence of such point-to-point exceptions, timing constraint violations may be worsened by retiming. Therefore, it is best not to apply retiming to designs with these exceptions. Case analysis constraints are also ignored when moving the sequential cells. The incremental compilation after moving the sequential cells takes all types of constraints into account.
- Designs cannot contain unmapped synthetic library components.

The movement of sequential cells and the handling of hierarchical subdesigns can be controlled as follows (in addition to the **-async_transform** and **-sync_transform** options):

- If a sequential cell has the **dont_touch** attribute set, **optimize_registers** does not move the sequential cell itself, nor does it move any other sequential cell across that cell.
- If instances contain the **dont_touch** attribute, they are not ungrouped. The **optimize_registers** command does not ungroup and does not place any registers inside an instance that has the **dont_touch** attribute set. If a hierarchical cell does not contain sequential cells and has the **dont_touch** attribute set, sequential cells can move across the cell. If the cell does contain sequential cells and has the **dont_touch** attribute set, sequential cells cannot move across the cell.
- The **optimize_registers** command can move sequential cells into a level of hierarchy. In this case, a clock pin is inserted into the interface of the instance if there was no clock pin previously. The new clock pin is named after the clock pin of the

enclosing hierarchical instance.

The **optimize_registers** command supports handling of black-box cells (that is, cells for which no timing is specified; for example, placeholders for RAMs). The **optimize_registers** command models a black-box cell as if the cell were external to the current design, without actually changing the interface of the design itself.

The **optimize_registers** command includes a delay modeling capability. For predictability, the algorithm selects from the target library a flip-flop or latch that has a small setup time, good drive at the outputs, and low input loading as compared with other flip-flops or latches in the library. This flip-flop or latch is designated as the preferred flip-flop or preferred latch. The preferred flip-flop/latch helps to provide tighter bounds on the performance variation after the **optimize_registers** sequence. To select a particular preferred flip-flop, use the **set_register_type** command. To exclude certain flip-flops, use the **set_dont_use** command.

The retiming process produces some delay report information. For details, see the *Design Compiler User Guide*. After retiming, implementation selection is no longer performed on synthetic library components in the design by subsequent executions of the **compile** command.

The **optimize_registers** command can also be used to optimize sub-critical paths using *delay threshold optimization*. Sub critical paths are those paths which operate at delays less than the final worst critical delay provided by retiming. Threshold optimization technique can be used to balance delays on paths with positive slacks. Optimize register uses target clock period specified using *delay_threshold* and performs multiple passes of min-period retiming on paths that operate at delays less than worst delay but greater than target clock period.

The **optimize_registers** command is also supported in multicorner-multimode (MCMM) technology, which is available with Design Compiler Graphical.

EXAMPLES

The following example shows **optimize_registers** being applied to a netlist:

```
prompt> read pre_retiming.db

/*
 * Delete dont_touch'es used for
 * hierarchical compile strategy.
 */

prompt> optimize_registers
```

SEE ALSO

current_design(2)
set_dont_touch(2)
set_dont_use(2)
set_transform_for_retiming(2)
set_register_type(2)
set_optimize_registers(2)
attributes(3)

parallel_execute

Runs reporting or checking commands in parallel.

SYNTAX

```
status parallel_execute  
  [-out file_name]  
  [-list_all]  
  [report_command_list]
```

Data Types

file_name string
report_command_list list

ARGUMENTS

-out *file_name*

Specifies the file name for the generated output file.

-list_all

Lists all supported commands that can run in parallel using this command.

report_command_list

Specifies a list of checking or reporting commands to execute in parallel.

DESCRIPTION

This command runs various supported reporting and checking commands in parallel. You must specify a list of commands to run. New jobs are launched to run the listed commands. The **parallel_execute** command completes after the longest running command has finished. Use the **set_host_options** command to specify the number of cores that the **parallel_execute** command can use to run the listed commands in parallel. The command can use only up to eight cores by default. When you specify more than eight cores, the tool overrides it with the default. You should run the **update_timing** command before issuing **parallel_execute** command.

You must enclose the complete command string with double quotation marks, for example, "**report_timing -input_pins -nets -path_type only**".

EXAMPLES

The following example shows a **parallel_execute** run.

```
prompt> update_timing
prompt> set_host_options -max_cores 4
prompt> parallel_execute [list "report_qor" \
    "report_timing -from a -to z2 -path_type short" \
    "report_cell -connections -verbose {t_bar}" \
    "report_design"]
```

The following example redirects the output of the command to a file. The tool handles write and append modes properly. Each command output can be redirected to a separate file or append mode can be used to append the output to an existing file.

```
prompt> set LOG "qor_report.log"
prompt> update_timing
prompt> set_host_options -max_cores 2
prompt> parallel_execute [list \
    "report_timing -input_pins -nets -path_type only > $LOG" \
    "report_qor -summary >> $LOG"]
```

The following example redirects the output of all commands to a file. Output of all the listed commands are written to the report.log file.

```
prompt> set LOG "report.log"
prompt> update_timing
prompt> set_host_options -max_cores 2
prompt> parallel_execute -out $LOG [list "report_timing" "report_qor" "report_cell"]
```

SEE ALSO

set_host_options(2)

parse_proc_arguments

Parses the arguments passed into a Tcl procedure.

SYNTAX

string **parse_proc_arguments**

ARGUMENTS

The **parse_proc_arguments** command has no arguments.

DESCRIPTION

The **parse_proc_arguments** command is used within a Tcl procedure to enable use of the -help option, and to support argument validation. It should typically be the first command called within a procedure. Procedures that use **parse_proc_arguments** will validate the semantics of the procedure arguments and generate the same syntax and semantic error messages as any application command (see the examples that follow).

When a procedure that uses **parse_proc_arguments** is invoked with the -help option, **parse_proc_arguments** will print help information (in the same style as using **help -verbose**) and will then cause the calling procedure to return. Similarly, if there was any type of error with the arguments (missing required arguments, invalid value, and so on), **parse_proc_arguments** will return a Tcl error and the calling procedure will terminate and return.

If you didn't specify -help, and the specified arguments were valid, the array variable *result_array* will contain each of the argument values, subscripted with the argument name. Note that the argument name here is NOT the names of the arguments in the procedure definition, but rather the names of the arguments as defined using the **define_proc_attributes** command.

The **parse_proc_arguments** command cannot be used outside of a procedure.

EXAMPLES

The following procedure shows how **parse_proc_arguments** is typically used. The *argHandler* procedure parses the arguments it receives. If the parse is successful, *argHandler* prints the options or values actually received.

```
proc argHandler args {  
    parse_proc_arguments -args $args results  
    foreach argname [array names results] {
```

```

    echo " $argname = $results($argname)"
  }
}
define_proc_attributes argHandler -info "argument processor" \
  -define_args \
  {{-Oos "oos help" AnOos one_of_string {required value_help {values {a b}}}}
  {-Int "int help" AnInt int optional}
  {-Float "float help" AFloat float optional}
  {-Bool "bool help" "" boolean optional}
  {-String "string help" AString string optional}
  {-List "list help" AList list optional}}
  {-IDup int dup AIDup int {optional merge_duplicates}}}}

```

Invoking argHandler with the -help option generates the following:

```

prompt> argHandler -help
Usage: argHandler  # argument processor
      -Oos AnOos      (oos help:
                        Values: a, b)
      [-Int AnInt]    (int help)
      [-Float AFloat] (float help)
      [-Bool]         (bool help)
      [-String AString] (string help)
      [-List AList]   (list help)

```

Invoking argHandler with an invalid option causes the following output (and a Tcl error):

```

prompt> argHandler -Int z
Error: value 'z' for option '-Int' not of type 'integer' (CMD-009)
Error: Required argument '-Oos' was not found (CMD-007)

```

Invoking argHandler with valid arguments generates the following:

```

prompt> argHandler -Int 6 -Oos a
-Oos = a
-Int = 6

```

SEE ALSO

```

define_proc_attributes(2)
help(2)
proc(2)

```

preview_dft

Previews, but does not implement, the test points, scan chains, and on-chip clocking control logic to be added to the current design.

SYNTAX

```
status preview_dft
  [-show object_types]
  [-test_points all]
  [-test_wrappers all]
  [-bsd cells | data_registers | instructions | tap | all]
  [-script]
  [-verbose]
```

Data Types

object_types list

ARGUMENTS

-show *object_types*

Reports information for the objects specified as values, in addition to the summary report the command produces by default. You must specify one or more of the following values for the *object_types* argument:

- The **bidirectionals** argument displays bidirectional port conditioning in scan shift. Reports the specified and resolved conditioning for each bidirectional port.
- The **cells** argument displays scan cells. Scan segments are distinguished by using keys that default to **s**. Scan link locations are marked by keys placed on the cells that precede them. Wire keys default to **w**; scan-out lockup keys default to **l**. You can change key defaults by using the **test_scan_segment_key**, **test_scan_link_so_lockup_key**, and **test_scan_link_wire_key** variables.
- The **qgates** argument is similar to the **cells** argument, except that the report also displays "(g)" annotations beside scan cells whose functional outputs have been gated to reduce power during scan shift. This functional gating feature is enabled by the **set_scan_suppress_toggling** command. For more information, see the man page.
- The **scan** argument displays hierarchical **set_scan_element** command attribute values. Although **set_scan_element** commands can be applied to hierarchical cells, leaf cells, library cells, and designs, the **preview_dft** command reports scan attribute values only for hierarchical cells and leaf cells. These reflect the impact of every **set_scan_element** command you apply to the design. Scan cells with true scan attributes are distinguished by using keys that default to **t**. You can change the default by using the **test_scan_true_key** environment variable.
- The **scan_clocks** argument displays scan clock domains. For multiplexed flip-flop scan styles, it displays the first cell or segment clocked by a new clock in each chain, and it shows the last cell or segment in the chain. An ordered triplet shows the scan clock name, trigger time, and whether the clock edge is rising or falling.

The **preview_dft** command does not distinguish capture time and launch time because these are the same for edge-triggered elements. Additional entries are generated when either the scan clock or trigger time changes. It displays exit and entry points. Lines containing three dots (...) show that there are intervening scan cells. Only entry points have ordered triplets, which are aligned for readability. If internal clocks have been created in the design by using the **set_scan_configuration** command with the **-internal_clocks** option or the **create_test_clock** command with the **-internal_clocks** option, the **preview_dft** command displays internal clock pin names instead of the external clock names. The triplet is replaced by "(No clock)" if the cell or segment does not have scan clock information.

For other scan styles, using the **preview_dft** command with the **-show_scan_clocks** option prints the scan clocks for every scan chain.

- The **scan_signals** argument displays scan signals, including hookup pins and sense values between them. If scan ports are not shared with mission mode ports, the **preview_dft** command generates interim port names by using the strings specified following environment variables, as appropriate, with %s characters deleted. The following names can change when the ports are created to avoid naming conflicts:

```
test_clock_port_naming_style
test_scan_clock_a_port_naming_style
test_scan_clock_b_port_naming_style
test_scan_clock_port_naming_style
```

- The **segments** argument displays scan segments. It also displays segment scan inputs, scan outputs, scan enables, and scan clocks.
- The **tristates** argument displays the disabling in scan shift for all tristate nets in a design. It reports the specified and resolved disabling for each tristate net.
- The **voltages** argument displays the voltage value for a scan cell or scan segment. It displays the operating voltages of the cell or segment in a scan chain. The voltage is displayed only for the first cell or segment and any other cell or segment at voltage boundaries.
- The **power_domains** argument displays the power domain ID for a scan cell or scan segment. It displays the power domain ID of the cell or segment in a scan chain. The power domain ID is displayed only for the first cell or segment, and any other cell or segment at power domain boundaries.
- The **scan_summary** argument displays scan chain information in short format.
- The **all** argument displays all information about a scan configuration.

The **-show** and **-script** options are mutually exclusive; do not use them together. If you attempt to use them together, the command generates a warning message, and the **-show** option is ignored.

-test_points all

Reports all test points information, in addition to the summary report the **preview_dft** command produces by default. The information displayed includes names assigned to the test points, locations of violations being fixed, names of test mode tags, logic states that enable the test mode, and names of data sources or sinks for the test points.

The **-test_points** and **-script** options are mutually exclusive; do not use them together. If you attempt to use them together, the command generates a warning message, and the **-test_points** option is ignored.

-test_wrappers all

Reports all information about core wrappers, in addition to the summary report the command produces by default. This option displays the number of interface ports, ports wrapped, type of wrapper cells inserted, and ports excluded from wrapping. You cannot use this option with the **-script** option.

-bsd cells | data_registers | instructions | tap | all

Specifies that **preview_dft** reports detailed information about one or more sections of the boundary-scan architecture. The valid

values are as follows:

- **cells** shows the boundary-scan register cells.
- **data_registers** shows the name and length of each boundary-scan test data register.
- **instructions** shows the boundary-scan instructions and their optimization codes.
- **tap** shows the TAP ports.
- **all** shows all details of the boundary-scan architecture.

-script

Produces a shell script, written to standard output, that specifies the required scan configuration. You cannot use this option with the **-show**, **-test_points**, or **-test_wrappers** option. If you attempt to use **-script** with these options, the command generates a warning message, and ignores the **-show**, **-test_points**, and **-test_wrappers** options.

-verbose

Generates multiple warning messages for a set of similar cells or pins with the same design rule violation. By default, a warning message is generated for only the first cell or pin found for a rule violation.

DESCRIPTION

This command previews, but does not implement, the test points, scan chains, and clock controllers to be added to the current design. The preview is based on the current set of scan, test point, and clock controller specifications. The command generates a report without actually performing any synthesis tasks. The report shows the effects of the current specifications, which you can modify, and then rerun the command as many times as needed until you are satisfied with the previewed results.

Process

The **preview_dft** command first generates information on the scan architecture that will be implemented. In the case of a DFTMAX insertion, **preview_dft** provides information about the compressor being created, and for basic scan, the specific chain information for the design.

Next, the command generates and displays a scan chain design that satisfies scan specifications on the current design. This design is exactly the scan chain design that is presented to the **insert_dft** command for synthesis, so you can preview your test point and scan chain designs without synthesizing the design; you can change your specifications to explore the design space, as necessary.

Executing the command without options generates a summary report that includes the number of test points, test_modes, test point enables, sources and sinks, number of scan chains, test methodology, and scan style. Applicable clock domain constraints, scan enable (or scan enable inverted), and hookup pins are included if the scan style is **multiplexed_flip_flop**. For each scan chain, the command displays the chain name, scan-in port, scan-out port, and length.

If scan ports are not shared with mission mode ports and new ports are created, the command generates interim port names. This is performed by appending an internal integer chain index to the strings specified in the following environment variables, as appropriate, with %s characters deleted. These names can change when the ports are created to avoid cell-naming conflicts.

```
test_scan_in_port_naming_style
test_scan_out_port_naming_style
test_scan_enable_port_naming_style
test_scan_enable_inverted_port_naming_style
```

EXAMPLES

The following example displays scan clocks:

prompt> **preview_dft -show scan_clocks**

Information: Using test design rule information from previous dft_drc run.

Architecting Scan Chains

Preview_dft report

For : 'Insert_dft' command

Design : top

Version: G-2012.06-SP2

Date : Wed Oct 10 12:32:05 2012

Number of chains: 1

Scan methodology: full_scan

Scan style: multiplexed_flip_flop

Clock domain: mix_clocks

(l) shows cell scan-out drives a lockup latch

(s) shows cell is a scan segment

(w) shows cell scan-out drives a wire

Scan chain '1' (SI1 --> SO1) contains 5 cells:

ENAB_reg (l) (CLK, 45.0, rising)

URX/DATA_reg[0] (RXCLK, 45.0, rising)

...

UTX/DATA_reg[0] (TXCLK, 45.0, rising)

UTX/DATA_reg[1]

The following example displays scan cells (with the report header omitted):

prompt> **preview_dft -show cells**

...

(l) shows cell scan-out drives a lockup latch

(s) shows cell is a scan segment

(w) shows cell scan-out drives a wire

Scan chain '1' (SI1 --> SO1) contains 5 cells:

ENAB_reg (l)

URX/DATA_reg[0]

URX/DATA_reg[1] (l)

UTX/DATA_reg[0]

UTX/DATA_reg[1]

The following example displays scan cells with functional output gating:

prompt> **preview_dft -show qgates**

...

- (l) shows cell scan-out drives a lockup latch
- (s) shows cell is a scan segment
- (w) shows cell scan-out drives a wire
- (g) shows cell scan-out drives a toggle suppressing gate

Scan chain '1' (SI1 --> SO1) contains 5 cells:

```
ENAB_reg (l)
URX/DATA_reg[0] (g)
URX/DATA_reg[1] (l) (g)
UTX/DATA_reg[0] (g)
UTX/DATA_reg[1] (g)
```

The following example shows the **preview_dft** command with the **-show** option set to **all**, reporting scan chain design information. The cell named **#1** is not added to the scan chain because it is running a **set_scan_element** command set to **false**.

prompt> **preview_dft -show all**

Information: Using test design rule information from previous dft_drc run.
Architecting Scan Chains

Preview_dft report

For : 'Insert_dft' command

Design : top

Version: G-2012.06-SP3

Date : Wed Oct 10 12:41:51 2012

Number of chains: 1

Scan methodology: full_scan

Scan style: multiplexed_flip_flop

Clock domain: mix_clocks

- (l) shows cell scan-out drives a lockup latch
- (s) shows cell is a scan segment
- (t) shows cell has a true scan attribute
- (w) shows cell scan-out drives a wire

Scan chain '1' (SI1 --> SO1) contains 5 cells:

```
ENAB_reg (l)          (CLK, 45.0, rising)
URX/DATA_reg[0]      (RXCLK, 45.0, rising)
URX/DATA_reg[1] (l)
UTX/DATA_reg[0]      (TXCLK, 45.0, rising)
UTX/DATA_reg[1]
```

Scan signals:

test_scan_in: SI1 (no hookup pin)
test_scan_out: SO1 (no hookup pin)

No user-defined segments

No multibit segments

No cells have scan true

1 cell has scan false:

RSTN_reg

No tristate nets.

No bidirectionals.

***** Test Point Plan Report *****

Total number of test points : 0

Number of Autofix test points: 0

Number of Wrapper test points: 0

Number of test modes : 0

Number of test point enables : 0

Number of data sources : 0

Number of data sinks : 0

1

The following example shows the results when using the **preview_dft** command with the **-script** option run on the same design:

```
prompt> preview_dft -script
#####
#                                     #
# Messages from preview_dft          #
# WARNING: Remove these messages before sourcing this script #
#                                     #
#####

# Information: Using test design rule information from previous dft_drc run.
#

#####
#                                     #
# Created by preview_dft() on Wed Oct 10 12:42:52 2012
#                                     #
#####

set_scan_path 1 -dedicated_scan_out false -test_mode Internal_scan -ordered_elements { \
"ENAB_reg" \
"URX/DATA_reg[0]" \
"URX/DATA_reg[1]" \
"UTX/DATA_reg[0]" \
"UTX/DATA_reg[1]" \
}

1
```

The following example shows the **preview_dft** command reporting test point and scan chain design information. The report also shows on-chip clocking controllers that will be added.

```
prompt> preview_dft -test_point all
Information: Using test design rule information from previous dft_drc run.
```

```

Running Autofix
Architecting Test Points
*****

PLL clock information preview
*****

*****

Cell name: my_pll_cntrl
Design name: snps_clk_mux
*****

Number of bits per clock: 2
Clock chain length: 2
Clock chain count: 1
Fast clock pins:
  my_pll/pll_clk
Slow clock ports/pins:
  CLK
Checking compatibility of PLL clock controller 'snps_clk_mux'
Architecting Scan Chains

```

```

*****
Preview_dft report
For : 'Insert_dft' command
Design : top
Version: G-2012.06-SP2
Date : Wed Oct 10 13:04:04 2012
*****

```

```

Number of chains: 1
Scan methodology: full_scan
Scan style: multiplexed_flip_flop
Clock domain: mix_clocks

```

Scan chain '1' (SI1 --> SO1) contains 5 cells

```

***** Test Point Plan Report *****
Total number of test points : 1
Number of Autofix test points: 1
Number of Wrapper test points: 0
Number of test modes      : 1
Number of test point enables : 0
Number of data sources     : 1
Number of data sinks       : 0
*****

```

```

Dft signals:
TestMode: TM (no hookup pin)
TestData: FIX (no hookup pin)

```

TEST POINTS

CLIENT	NAME	TYPE	LOCATIONS MODES	TEST POINT ENABLE	TEST SOURCE/ SINK	DATA

```
Autofix  test_point F-1  URX/DATA_reg[0]/RN  TM      handler  FIX
          URX/DATA_reg[1]/RN
          UTX/DATA_reg[0]/RN
          UTX/DATA_reg[1]/RN
```

TEST MODES

TAG	NEW/ EXISTING	TYPE	NAME	CLOCK
TM	Existing	Port	TM	none

TEST POINT ENABLES

TAG	NEW/ EXISTING	TYPE	NAME	CLOCK
handler	unknown	Logic 1	unknown	none

DATA SOURCES

TAG	NEW/ EXISTING	TYPE	NAME	CLOCK
FIX	Existing	Port	FIX	none

1

The following example displays a scan summary:

```
prompt> preview_dft-show scan_summary
```

```
Information: Using test design rule information from previous dft_drc run.
Architecting Scan Chains
```

```
*****
```

```
Preview_dft report
```

```
For : 'Insert_dft' command
```

```
Design : top
```

```
Version: G-2012.06-SP3
```

```
Date : Wed Oct 10 12:43:38 2012
```

```
*****
```

```
Number of chains: 1
```

```
Scan methodology: full_scan
```

```
Scan style: multiplexed_flip_flop
```

```
Clock domain: mix_clocks
```

Chain	Scan Ports (si --> so)	# of Cells	Inst/Chain	Clock (port, time, edge)
S 1	SI1 --> SO1	5	ENAB_reg (I)	(CLK, 45.0, rising)
			URX/DATA_reg[0]	(RXCLK, 45.0, rising)
			UTX/DATA_reg[0]	(TXCLK, 45.0, rising)

1

SEE ALSO

current_design(2)
insert_dft(2)
set_dft_configuration(2)
set_dft_signal(2)
set_dont_touch(2)
set_scan_configuration(2)
set_scan_element(2)
set_scan_link(2)
set_scan_path(2)
set_scan_suppress_toggling(2)
write(2)
target_library(3)
test_scan_link_so_lockup_key(3)
test_scan_link_wire_key(3)
test_scan_segment_key(3)

print_message_info

Prints information about diagnostic messages that have occurred or have been limited.

SYNTAX

```
string print_message_info  
    [-ids id_list]  
    [-summary]
```

Data Types

id_list list

ARGUMENTS

-ids *id_list*

Specifies a list of message identifiers to report. Each entry can be a specific message or a glob-style pattern that matches one or more messages. If this option is omitted and no other options are given, then all messages that have occurred or have been limited are reported.

-summary

Generates a summary of error, warning, and informational messages that have occurred so far.

DESCRIPTION

The **print_message_info** command enables you to print summary information about error, warning, and informational messages that have occurred or have been limited with the **set_message_info** command. For example, if the following message is generated, information about it is recorded:

```
Error: unknown command 'wrong_command' (CMD-005)
```

It is useful to be able to summarize all recorded information about generated diagnostic messages. Much of this can be done using the **get_message_info** command, but you need to know a specific message ID. By default, **print_message_info** summarizes all of the information. It provides a single line for each message that has occurred or has been limited, and one summary line that shows the total number of errors, warnings, and informational messages that have occurred so far. If an *id_list* is given, then only messages matching those patterns are displayed. If **-summary** is given, then a summary is displayed.

Using a pattern in the *id_list* is intended to show a specific message prefix, for example, "CMD*". Note that this does not show all messages with that prefix. It shows only the messages that have occurred or have been limited.

EXAMPLES

The following example uses **print_message_info** to show a few specific messages:

```
prompt> print_message_info -ids [list "CMD*" APP-99]
```

Id	Limit	Occurrences	Suppressed
CMD-005	0	7	2
APP-99	1	0	0

At the end of the session, you might want to generate some information about a set of interesting messages, such as how many times each occurred (which includes suppressions), how many times each was suppressed, and whether a limit was set for any of them. The following example uses **print_message_info** to get this information:

```
prompt> print_message_info
```

Id	Limit	Occurrences	Suppressed
CMD-005	0	12	0
APP-027	100	150	50
APP-99	0	1	0

Diagnostics summary: 12 errors, 150 warnings, 1 informational

Note that the suppressed count is not necessarily the difference between the limit and the occurrences, since the limit can be dynamically changed with the **set_message_info** command.

SEE ALSO

`get_message_info(2)`
`set_message_info(2)`
`suppress_message(2)`

print_suppressed_messages

Displays an alphabetical list of message IDs that are currently suppressed.

SYNTAX

string **print_suppressed_messages**

ARGUMENTS

The **print_suppressed_messages** command has no arguments.

DESCRIPTION

The **print_suppressed_messages** command displays all messages that you suppressed using the **suppress_message** command. The messages are listed in alphabetical order. You only can suppress informational and warning messages. The result of **print_suppressed_messages** is always the empty string.

EXAMPLES

The following example shows the output from the **print_suppressed_messages** command:

```
prompt> print_suppressed_messages  
No messages are suppressed
```

```
prompt> suppress_message {XYZ-001 CMD-029 UI-1}
```

```
prompt> print_suppressed_messages  
The following 3 messages are suppressed:  
CMD-029, UI-1, XYZ-001
```

SEE ALSO

suppress_message(2)
unsuppress_message(2)

print_variable_group

Lists the variables defined in a specified variable group, along with their current values.

SYNTAX

```
status print_variable_group
      group
```

Data Types

group string

ARGUMENTS

group

Specifies the name of the group of variables for which you want to see the definitions and current values.

DESCRIPTION

The **print_variable_group** command lists the variables and related values defined in *group*. The current values for *group* are as follows:

all	All variables
acs	Variables that affect Automated Chip Synthesis
bsd	Variables that affect the insert_dft , optimize_bsd , check_bsd , and write_bsd commands
compile	compile command variables
dpcm	Variables that affect DPCM
hdl	Variables that affect reading, writing, and optimizing HDL
insert_dft	insert_dft command variables
io	read_file , read_lib , and write command variables
links_to_layout	links_to_layout variables
multibit	Variables that affect compile multibit mapping
power	Variables that affect Power Compiler commands
preview_scan	Variables that affect the preview_scan command
suffix	Variables that define file suffixes
synlib	cache_ls command variables
system	Global system variables
test	DFT Compiler variables
testmanager	Variables that affect fault list handling

timing	Variables that affect timing
vhdlIo	Variables that affect writing VHDL files and libraries
write_test	write_test command variables

The available variable groups are configurable. Change variable groups or create new ones with the **group_variable** command.

EXAMPLES

The following example displays the variables in the *timing* variables group, along with their current values:

```
prompt> print_variable_group timing
case_analysis_log_file = ""
case_analysis_sequential_propagate = "false"
case_analysis_with_logic_constants = "false"
create_clock_no_input_delay = "false"
disable_auto_time_borrow = "false"
disable_case_analysis = "false"
disable_conditional_mode_analysis = "false"
disable_library_transition_degradation = "false"
.
.
.
timing_self_loops_no_skew = "false"
when_analysis_permitted = "true"
when_analysis_without_case_analysis = "false"
1
```

SEE ALSO

dc_shell(1)
group_variable(2)

printenv

Prints the value of environment variables.

SYNTAX

```
string printenv  
[variable_name]
```

Data Types

```
variable_name  string
```

ARGUMENTS

variable_name

Specifies the name of a single environment variable to print.

DESCRIPTION

The **printenv** command prints the values of the environment variables inherited from the parent process or set from within the application with the **setenv** command. When no arguments are specified, all environment variables are listed. When a single *variable_name* is specified, if that variable exists, its value is printed. There is no useful return value from **printenv**.

To retrieve the value of a single environment variable, use the **getenv** command.

EXAMPLES

The following examples show the output from the **printenv** command:

```
prompt> printenv SHELL  
/bin/csh
```

```
prompt> printenv EDITOR  
emacs
```


SEE ALSO

getenv(2)

setenv(2)

printvar

Prints the values of one or more variables.

SYNTAX

```
string printvar  
    [pattern]  
    [-user_defined | -application]
```

Data Types

pattern string

ARGUMENTS

pattern

Prints variable names that match *pattern*. The optional *pattern* argument can include the wildcard characters * (asterisk) and ? (question mark). If not specified, all variables are printed.

-user_defined

Shows only user-defined variables. This option is mutually exclusive with the **-application** option.

-application

Shows only application variables. This option is mutually exclusive with the **-user_defined** option.

DESCRIPTION

The **printvar** command prints the values of one or more variables. If the *pattern* argument is not specified, the command prints out the values of application and user-defined variables. The **-user_defined** option limits the variables to those that you have defined. The **-application** option limits the variables to those created by the application. The two options are mutually exclusive and cannot be used together.

Some variables are suppressed from the output of **printvar**. For example, the Tcl environment variable array *env* is not shown. To see the environment variables, use the **printenv** command. Also, the Tcl variable **errorInfo** is not shown. To see the error stack, use the **error_info** command.

EXAMPLES

The following command prints the values of all of the variables:

```
prompt> printvar
```

The following command prints the values of all application variables that match the pattern *sh*a**:

```
prompt> printvar -application sh*a*  
sh_arch          = "sparcOS5"  
sh_command_abbrev_mode = "Anywhere"  
sh_enable_page_mode  = "false"  
sh_new_variable_message = "false"  
sh_new_variable_message_in_proc = "false"  
sh_source_uses_search_path = "false"
```

The following command prints the value of the **search_path** variable:

```
prompt> printvar search_path  
search_path      = ". /designs/newcpu/v1.6 /lib/cmos"
```

The following command prints the values of the user-defined variables:

```
prompt> printvar -user_defined  
a                = "6"  
designDir         = "/u/designs"
```

SEE ALSO

error_info(2)
printenv(2)
setenv(2)

proc_args

Displays the formal parameters of a procedure.

SYNTAX

```
string proc_args  
      proc_name
```

Data Types

```
proc_name  string
```

ARGUMENTS

proc_name

Specifies the name of the procedure.

DESCRIPTION

The **proc_args** command is used to display the names of the formal parameters of a user-defined procedure. This command is essentially a synonym for the Tcl builtin command **info** with the *args* argument.

EXAMPLES

This example shows the output of **proc_args** for a simple procedure:

```
prompt> proc plus {a b} { return [expr $a + $b] }
```

```
prompt> proc_args plus  
a b
```

```
prompt> info args plus  
a b
```

```
prompt>
```

SEE ALSO

info(2)
proc(2)
proc_body(2)

proc_body

Displays the body of a procedure.

SYNTAX

```
string proc_body  
      proc_name
```

Data Types

```
proc_name  string
```

ARGUMENTS

proc_name

Specifies the name of the procedure.

DESCRIPTION

The **proc_body** command is used to display the body (contents) of a user-defined procedure. This command is essentially a synonym for the Tcl builtin command **info** with the *body* argument.

EXAMPLES

This example shows the output of **proc_body** for a simple procedure:

```
prompt> proc plus {a b} { return [expr $a + $b] }
```

```
prompt> proc_body plus  
return [expr $a + $ b]
```

```
prompt> info body plus  
return [expr $a + $ b]
```

```
prompt>
```

SEE ALSO

info(2)
proc(2)
proc_args(2)

propagate_constraints

Propagates timing constraints from lower levels of the design hierarchy to the current design.

SYNTAX

```
status propagate_constraints
  [-design design_list]
  [-all]
  [-clocks]
  [-disable_timing]
  [-dont_apply]
  [-false_path]
  [-gate_clock]
  [-ideal_network]
  [-ignore_from_or_to_port_exceptions]
  [-ignore_through_port_exceptions]
  [-max_delay]
  [-min_delay]
  [-multicycle_path]
  [-operating_conditions]
  [-power_supply_data]
  [-output file_name]
  [-port_isolation]
  [-verbose]
  [-case_analysis]
  [-target_library_subset]
  [-opcond_inference]
```

Data Types

design_list list
file_name string

ARGUMENTS

-design *design_list*

Specifies a list of names of lower-level designs from which constraints are to be propagated. The constraints are propagated from every instance of the designs in this list. By default, constraints are propagated from every lower-level design in the hierarchy.

-all

Propagates clocks, **gate_clock** checks, **false_path** exceptions, **multicycle_path** exceptions, **max_delay** and **min_delay** exceptions, instance-specific operating conditions, power supply data, and **disable_timing** attributes. This is the default. Issuing

the command without options has the same effect as specifying the **-all** option.

-clocks

Propagates only clocks previously created by using the **create_clock** command. Virtual clocks are not propagated.

-disable_timing

Propagates only **disable_timing** attributes previously set by using the **set_disable_timing** command.

-dont_apply

Prevents the propagated constraints from being applied to the current design. Warnings are still reported. You can use this option with the **-verbose** option to see the effects of propagating the constraints without actually propagating them. If you do not use the **-verbose** option, the command only reports warnings about conflicting exceptions.

-false_path

Propagates only **false_path** exceptions previously specified by using the **set_false_path** command.

-gate_clock

Note: This option will be obsolete in a future release. Clock-gating insertion with **compile_ultra -gate_clock** automatically propagates clock gate setup and hold time.

Propagates only setup and hold clock gating checks previously specified by using the **set_clock_gating_check** command. RTL clock gating automatically imposes setup and hold time constraints during elaboration.

-ideal_network

Propagates ideal networks previously created by using the **set_ideal_network** command.

-ignore_from_or_to_port_exceptions

Propagates exceptions for ports on *from_lists* and/or *to_lists* of commands that set timing constraints. By default, the command converts "from port" and "to port" exceptions into "through" exceptions and propagates them to the current design.

-ignore_through_port_exceptions

Prevents propagation of exceptions for ports on *through_lists* of commands that set timing constraints. By default, these exceptions are propagated.

-max_delay

Propagates only **max_delay** exceptions previously specified by using the **set_max_delay** command.

-min_delay

Propagates only **min_delay** exceptions previously specified by using the **set_min_delay** command.

-multicycle_path

Propagates only **multicycle_path** exceptions previously specified by using the **set_multicycle_path** command.

-operating_conditions

Propagates only **operating_conditions** previously specified by using the **set_operating_condition** command. It propagates the operating conditions specified on instances as well as those specified on lower-level subdesigns.

-power_supply_data

Propagates only power intent data, including UPF (supply network, power state table, and various power intent strategies) and operating voltages on the supply nets.

-output *file_name*

Writes the command output to the file specified by *file_name*.

-port_isolation

Propagates exceptions previously set by the **set_isolate_ports** command.

-verbose

Reports equivalent constraint-setting commands that are applied to the current design as a result of using the **propagate_constraints** command. Following each propagated constraint, the name enclosed in /* */ is the name of the lower-level design from which the constraint is propagated.

-case_analysis

Propagates case values previously set by using the **set_case_analysis** command.

-target_library_subset

Propagates target library subsets that have been specified by using the **set_target_library_subset** command. It propagates the target library subset specified on instances as well as those specified on subdesigns.

-opcond_inference

Propagates only the strategies of operating condition inference that have been specified by using the **set_opcond_inference** command. It propagates the strategies specified on instances as well as those specified on subdesigns.

DESCRIPTION

This command collects constraints from instances of lower-level designs and applies those constraints to the current design for use during timing analysis and compilation. After the constraints are propagated, you can remove, override, or report them like any other constraints that are set on the current design. Issuing the command without options is the same as issuing it with the **-all** option.

The command produces a warning in some cases where it does not propagate constraints because of one of the following conditions:

- Clock name conflict: multiple clocks from different designs have the same name.
- Clock source conflict: an object is specified as a clock source of more than one clock.
- Clock exception from/to unpropagated clock: if a clock is not propagated, exceptions from or to the clock are not propagated. The command does not propagate a clock if you do not use the **-clock** option, if the clock is a virtual clock, or if propagating the clock creates a conflict.

The command does not propagate a constraint that creates a conflict; the constraint is echoed in comments using the object names in the current design.

You must propagate timing **gated_clock** checks before compilation if automatic clock gating has been performed during elaboration.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example propagates only false path constraints, from all lower-level designs to the current design:

```
prompt> propagate_constraints -false_path
```

The following example propagates only **multicycle_path** exceptions from the SUB1 subdesign to the current design:

```
prompt> propagate_constraints -design SUB1 -multicycle_path
```

The following example propagates clocks, exceptions, **gated_clock** checks, and **disable_timing** arcs to the current design from all instances of lower-level designs. The output is shown in non-verbose mode. Clock warnings are issued in this mode.

```
prompt> propagate_constraints -all
```

```
/******

Created by propagate_constraints() on Thu Jun 24 06:02:01 1999

*****/

/* Propagate Constraints from cell mid_left/ (MID_LEFT) */

/* Propagate Constraints from cell mid_right/ (MID_RIGHT) */

/* Propagate Constraints from cell mid_right/in_left/ (IN_LEFT) */

/* Propagate Constraints from cell mid_right/in_right/ (IN_RIGHT) */

/** Warning: Clock clk was already found in design: OUTIE */
/** The following clock definition in design IN_RIGHT */
/** is ignored :*/
/* create_clock -period 10 -waveform {0 5} (get_ports "clk") */
/** Warning: clock exception(s) in design IN_RIGHT is(are) ignored */
/** because clock clk_virtual1 was not propagated */
/** Warning: The following clock exceptions in design IN_RIGHT are ignored */
/* set_false_path */
/* -from (get_clocks "clk_virtual1") */
/* -to (get_pins "mid_right/in_right/F1/D") */
*/ /* IN_RIGHT */
1
```

The following example displays all constraints that would have been propagated by the command, but does not apply them to the current design. The output is shown in verbose mode.

```
prompt> propagate_constraints -all -verbose -dont_apply
```

```
*****/

Created by propagate_constraints() on Thu Jun 24 06:02:02 1999
*****/

/* Propagate Constraints from cell mid_left/ (MID_LEFT) */

/** Warning: clock exception(s) in design MID_LEFT is(are) ignored */
/** because clock virtual_clkm was not propagated */
/** Warning: The following clock exceptions in design MID_LEFT are ignored */
/* set_false_path
```

```

    -to (get_clocks "virtual_clk")
*/ /* MID_LEFT */

/* Propagate Constraints from cell mid_right/ (MID_RIGHT) */

set_false_path
  -from (get_pins "mid_right/in_left/F2/CP")
  -to (get_pins "mid_right/in_right/F2/D")
/* MID_RIGHT */
set_disable_timing (get_pins "mid_right/in_right/F2/Q") /* MID_RIGHT */

/* Propagate Constraints from cell mid_right/in_left/ (IN_LEFT) */

create_clock -name "clkin_l" -period 10
-waveform {0 5} (get_pins
"mid_right/in_left/clock") /* IN_LEFT */
set_false_path
  -to (get_clocks "clkin_l")
/* IN_LEFT */
set_false_path
  -through {(get_pins "mid_right/in_left/i1")
get_pins "mid_right/in_left/clock"}}

  -to (get_clocks "clkin_l")
/* IN_LEFT */
set_false_path
  -through (get_pins "mid_right/in_left/i1")
  -to (get_pins "mid_right/in_left/F1/D")
/* IN_LEFT */
set_disable_timing (get_cells "mid_right/in_left/F1") /* IN_LEFT */
set_disable_timing (get_cells "mid_right/in_left/F2")
-from "CP" -to "Q" /* IN_LEFT */

/* Propagate Constraints from cell mid_right/in_right/ (IN_RIGHT) */

/** Warning: Clock clk was already found in design: OUTIE */
/** The following clock definition in design IN_RIGHT*/
/** is ignored */
/* create_clock -period 10 -waveform {0 5} (get_ports "clk")
*/
/** Warning: clock exception(s) in design IN_RIGHT is(are) ignored */
/** because clock clk_virtual1 was not propagated */
/** Warning: The following clock exceptions in design IN_RIGHT are ignored */
/* set_false_path
  -from (get_clocks "clk_virtual1")
  -to (get_pins "mid_right/in_right/F1/D")
*/ /* IN_RIGHT */
1

```

SEE ALSO

`create_clock(2)`

`propagate_constraints`

```
report_timing_requirements(2)
set_clock_gating_check(2)
set_clock_gating_style(2)
set_disable_timing(2)
set_false_path(2)
set_ideal_network(2)
set_max_delay(2)
set_min_delay(2)
set_multicycle_path(2)
set_operating_conditions(2)
set_opcond_inference(2)
set_target_library_subset(2)
```

propagate_switching_activity

Forces the propagation of power-switching activity information.

SYNTAX

```
status propagate_switching_activity  
  [-verbose]  
  [-infer_related_clocks]
```

ARGUMENTS

-verbose

Specifies that the switching activity mechanism is run in verbose mode, which displays detailed information and warning messages.

-infer_related_clocks

Specifies that only related clock information is to be inferred. When this option is used, static probability and toggle rate information is not propagated; but the related clock information is inferred for design objects that request a related clock.

DESCRIPTION

Use of the **propagate_switching_activity** command is not recommended. This feature will be obsolete in a future release. The **propagate_switching_activity** command forces a propagation of the power switching activity information.

During the propagation of the switching activity, the information annotated using the **read_saif**, **set_switching_activity**, **merge_saif** commands is used to estimate the activity on unannotated nets. All commands that need switching activity information on all design objects use the switching activity propagation mechanism automatically. In most cases, you do not need to explicitly run the **propagate_switching_activity** command. The **propagate_switching_activity** command allows you to access propagated switching activity values directly without using a command such as the **report_power** command, which propagates the switching activity as a side effect.

The mechanism to propagate switching activity uses the static probability and toggle rate information you annotate to estimate the static probability and toggle rate information on unannotated design objects.

The mechanism used by the tool is based on a stochastic simulation method, which is that random activity based on the annotated switching activity is generated on the annotated design objects, and this activity is propagated across the design using a zero-delay simulator. This is repeated a few thousand times, depending on the effort level.

The **-infer_related_clocks** option is used to restrict the action of the **propagate_switching_activity** command to inferring only the related clock information on design objects on which it was requested. Use the **-clock ""** option of the **set_switching_activity**

command to specify that the related clock on the specified design object is inferred automatically by the tool. Related clock information is inferred when commands that need switching activity information (such as **report_power**) are used. Use the command with the **-infer_related_clocks** option to infer only the related clock information. The related clock is placed in the **related_clock** attribute on the design objects.

The method in which the related clock information is inferred is described in the following steps:

First, the tool determines a starting set of inferred related clocks using the following rules:

- The inferred related clock on an object with a user-set related clock, is the user-set related clock.
- The inferred related clock on a clock source port or pin is the clock.
- If the **power_rclock_inputs_use_clocks_fanout** variable is set to **true**, primary input nets that do not have an inferred related clock according to the rules already mentioned, the following rule is applied. If the transitive fanout of the input contains a clock network object, then the inferred related clock on the input is the clock driving the network. A clock network is the transitive fanout of a clock port or pin, stopping and including sequential cells. If more than one of this type of a clock object exists, then the fastest clock of this type is chosen as the related clock.

The related clocks are then inferred on design objects by the following rules:

- For flip-flops, if the **power_rclock_use_async_inputs** variable is set to **false**, then the inferred related clock on the cell's outputs is the inferred related clock on the cell's clock pin. If **power_rclock_use_async_inputs** is set to **true**, then the inferred related clock on the cell's outputs is the fastest inferred related clock on the cell's clock pin and the cell's asynchronous input pins.
- The inferred related clock on clock-gating cell outputs is the inferred related clock on the cell's clock input pin.
- For non-flip-flop and non-clock-gating cells, the inferred related clock on the cell's outputs is the fastest related clock on the cell's inputs.

The inferred related clocks are then made to be the design object's related clock for objects that requested a related clock. The inferred related clock information on design objects that did not request a related clock is discarded. For design objects that requested a related clock, but no clock was inferred by the above rules, the fastest design clock is set as the related clock if the **power_rclock_unrelated_use_fastest** variable is set to **true**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example propagates the power-switching activity information:

```
prompt> propagate_switching_activity
```

SEE ALSO

```
merge_saif(2)
read_saif(2)
report_saif(2)
reset_switching_activity(2)
```

```
set_switching_activity(2)
power_rclock_inputs_use_clocks_fanout(3)
power_rclock_unrelated_use_fastest(3)
power_rclock_use_asynch_inputs(3)
```

propagate_user_attributes

Propagates user attributes from lower levels of the design hierarchy to the current design.

SYNTAX

```
status propagate_user_attributes
  [-design design_list]
  [-verbose]
  attribute_list
```

Data Types

```
design_list    list
attribute_list list
```

ARGUMENTS

-design *design_list*

Specifies a list of names of lower-level designs from which the specified attributes are to be propagated. The specified attributes are propagated from every instance of the designs in this list. By default, the specified attributes are propagated from all lower-level designs in the hierarchy.

-verbose

Reports the lower-level designs that attributes are propagated from.

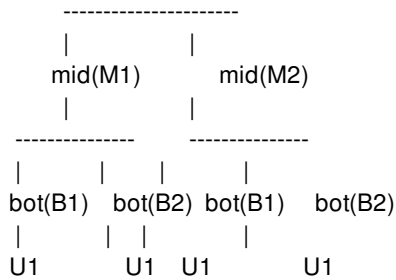
DESCRIPTION

This command propagates the specified user attributes from instances of lower-level designs and sets them on instances in the current design.

EXAMPLES

The following is an example of the **propagate_user_attributes** command. Assume you have the following design:

```
top
|
```



If you set the current design to mid and set the **set_user_attribute** value to XYZ on all cells in the mid design, when the current design is changed to top, the attributes set on the B1/U1 and B2/U1 instance cells inside mid are not propagated or visible from the top design. When you use the **propagate_user_attributes** command, the specified attributes are propagated to the instances in the current design, as shown in the following example.

```

prompt> current_design mid
Current design is 'mid'.
{mid}
prompt> define_user_attribute -type boolean -classes cell XYZ
XYZ
prompt> set_user_attribute [get_cells -hierarchy *] XYZ false
Information: Attribute 'XYZ' is set on 4 objects. (UID-186)
B1/U1 B2/U1 B1 B2
prompt> current_design top
Current design is 'top'.
{top}
prompt> get_cells -hierarchical -filter "XYZ==false"
{M1/B1 M1/B2 M2/B1 M2/B2}
prompt> propagate_user_attributes {XYZ}
1
prompt> get_cells -hierarchical -filter "XYZ==false"
{M1/B1/U1 M1/B2/U1 M1/B1 M1/B2 M2/B1 M2/B2}

```

SEE ALSO

define_user_attribute(2)
set_user_attribute(2)

push_down_model

Creates a new hierarchy around the current design for use with wrapping models flow in SoCTest.

SYNTAX

```
status push_down_model  
      new_design_name
```

Data Types

```
new_design_name  string
```

ARGUMENTS

new_design_name

Specifies the name of the new design to be created.

DESCRIPTION

The **push_down_model** command creates a new level of hierarchy around the current design. It creates a new design with the specified design name, instantiates the current design as an instance in the new design, creates ports in the new design for all corresponding pins of the instantiated cell, and connects all pins of the instantiated cell to the corresponding ports of the new design. The current design is set to the new design created by this command.

EXAMPLES

The following example creates a new design named *sub1_pushed* around the current design named *sub1*:

```
prompt> current_design sub1  
{sub1}
```

```
prompt> push_down_model sub1_pushed  
Current design is 'sub1_pushed'.
```

SEE ALSO

`insert_dft(2)`
`preview_dft(2)`

pwd

Displays the path name of the present working directory (pwd), also called the current directory.

SYNTAX

string **pwd**

ARGUMENTS

The **pwd** command has no arguments.

DESCRIPTION

The **pwd** command displays the path name of the current directory. The directory from which you invoke the shell is the initial current directory.

A file specification of dot (.) is shorthand for the current directory. Dot is commonly included in the **search_path** variable.

Use the **cd** command to change the current directory.

EXAMPLES

The following is an example of **pwd** showing the current working directory:

```
prompt> pwd  
/usr/designer/joe
```

SEE ALSO

cd(2)
sh(2)
search_path(3)

query_cell_instances

Finds all instances of a given library cell or module within the active scope. This is a UPF query command.

SYNTAX

```
list query_cell_instances
  cell_name
  [-domain domain_name]
```

Data Types

```
cell_name    string
domain_name  string
```

ARGUMENTS

cell_name

Specifies the name of the library cell or module to find in the active scope. The command reports all instances of the specified cell or module.

-domain *domain_name*

Limits the search to the instances in the specified power domain.

DESCRIPTION

The **query_cell_instances** command finds all instances of a mapped cell within the active scope. It returns a list of the found instances.

You can use this command at the tool shell prompt or in a script run by the **source** command, but not in a script run by the **load_upf** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports all instances of the AND2 library cell in the I_REG_FILE/PD_ALU power domain.

```
prompt> query_cell_instances -domain I_REG_FILE/PD_ALU AND2
I_ALU/C132 I_ALU/C133 I_ALU/C134 I_ALU/C135 I_ALU/C136
```

SEE ALSO

get_cells(2)
query_cell_mapped(2)

query_cell_mapped

Identifies the library cell or module mapped to a given instance. This is a UPF query command.

SYNTAX

```
string query_cell_mapped  
    instance_name
```

Data Types

```
instance_name  string
```

ARGUMENTS

instance_name

Specifies the name of the instance to query.

DESCRIPTION

The **query_cell_mapped** command identifies the library cell or module that is mapped to a given instance. It returns the cell name.

You can use this command at the tool shell prompt or in a script run by the **source** command, but not in a script run by the **load_upf** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the library cell mapped to instance I_ALU/C132.

```
prompt> query_cell_mapped I_ALU/C132  
AND2
```


SEE ALSO

`get_cells(2)`
`query_cell_instances(2)`

query_map_power_switch

Returns information about the previous mapping of a library cell to a power switch in the active scope. This is a UPF query command.

SYNTAX

```
list query_map_power_switch
    switch_name
    [-detailed]
```

Data Types

switch_name string

ARGUMENTS

switch_name

Specifies the name of the power switch. The command returns the full **map_power_switch** command previously used to map a library cell to the power switch.

-detailed

Returns the **map_power_switch** command information in the form of a Tcl list of keyword and value pairs.

DESCRIPTION

The **query_map_power_switch** command returns the full **map_power_switch** command previously used to map library cells to a specified power switch. The **-detailed** option returns the same information in the form of a Tcl list of keyword and value pairs. If you specify an asterisk for the switch name, the command returns the mapping information for all power switches. It returns a warning message for each switch that is not yet mapped.

You can use this command at the tool shell prompt or in a script run by the **source** command, but not in a script run by the **load_upf** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples show how to report the **map_power_switch** command previously used to map a library cell to power switch SW1.

```
prompt> query_power_switch *  
{SW1 SW2 }
```

```
prompt> query_map_power_switch SW1  
map_power_switch SW1 -domain TOP -lib_cells {STN_SWITCH_Y2_PM }
```

```
prompt> query_map_power_switch SW1 -detailed  
{switch_name SW1} {domain TOP} {lib_cells {STN_SWITCH_Y2_PM }}
```

```
prompt> query_map_power_switch SW2  
Warning: Can't find map_power_switch constraint for power switch SW2. (UPF-299)
```

SEE ALSO

```
map_power_switch(2)  
query_power_switch(2)
```

query_net_ports

Finds all the ports that are logically connected to a given net. This is a UPF query command.

SYNTAX

```
list query_net_ports
  net_name
  [-transitive true | false]
  [-leaf]
```

Data Types

net_name string

ARGUMENTS

net_name

Specifies the net. The command finds all ports connected to that net.

-transitive true | false

Specifies whether to search the only the current level of hierarchy (**false**, the default behavior) or both the current level and hierarchical descendants below the current level (**true**).

-leaf

Returns only the leaf-cell ports connected to the specified net. By default, both hierarchical and leaf-level cell ports are reported.

DESCRIPTION

The **query_net_ports** command returns the ports that are logically connected to the specified net. It returns a list of the port names. If no ports are connected to the specified net, it returns the null string.

You can use this command at the tool shell prompt or in a script run by the **source** command, but not in a script run by the **load_upf** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the ports connected to net I[30] at the current level of hierarchy.

```
prompt> query_net_ports I[30]  
I[30] I_LAT/Inst_1[30]
```

The following example reports the ports connected to net I[30] at the current level of hierarchy and in descendants below the current level.

```
prompt> query_net_ports I[30] -transitive true  
I_LAT/Inst_1_reg[30]/Q I[30] I_LAT/Inst_1[30]
```

The following example reports the ports of leaf-level cells connected to net I[30] at the current level of hierarchy and in descendants below the current level.

```
prompt> query_net_ports I[30] -transitive true -leaf  
I_LAT/Inst_1_reg[30]/Q
```

SEE ALSO

get_nets(2)
get_ports(2)
query_port_net(2)

query_objects

Searches for and displays objects in the database.

SYNTAX

```
string query_objects  
  [-verbose]  
  [-truncate elem_count]  
  [-class class_name]  
  object_spec
```

Data Types

<i>elem_count</i>	int
<i>class_name</i>	string
<i>object_spec</i>	list

ARGUMENTS

-verbose

Displays the class of each object found.

By default, the command lists only the name of each object. When you use this option, the command lists prefixes the name of each object with its class, as shown in the EXAMPLES section.

-truncate *elem_count*

Truncates the display to *elem_count* elements.

By default, the command displays up to 100 elements. To change the number of elements displayed by the command, use this option. To see all elements, set the *elem_count* value to 0.

-class *class_name*

Specifies the class used to search for objects when an element in the *object_spec* argument is not a collection. Valid classes are application-specific.

If you do not specify this option, the command displays only those elements in the *object_spec* argument that are collections and displays an error message for any other elements, as shown in the EXAMPLES section.

object_spec

Specifies the objects to find and display. Each element in the list is either a collection or a pattern.

- When you specify a collection, the command displays the contents of the collection.

- When you specify a pattern, the command searches the database for objects of the class specified in the **-class** option. If you do not specify the **-class** option, the command issues an error message for the pattern.

DESCRIPTION

The **query_objects** command finds and displays objects in the application's runtime database. The command does not have a meaningful return value; it displays the objects found and returns an empty string.

To control the number of elements displayed, use the **-truncate** option. If the display is truncated, the command prints an ellipsis (...) as the last element. If the default truncation occurs, the command displays a message that shows the total number of elements that would have displayed, as shown in the EXAMPLES section.

Although the output from the **query_objects** command looks similar to the output from any command that creates a collection, the **query_objects** command always returns an empty string.

The **query_objects** command displays the output in either a Legacy format or in a Tcl compatible format. The default output format varies by tool but you can control the format yourself by setting the **query_objects_format** variable. For example, to force Tcl-compatible output use this command:

```
prompt> set_app_var query_objects_format Tcl
Tcl
```

The EXAMPLES section shows output in both the Legacy and Tcl formats.

EXAMPLES

These following examples show the basic usage of the **query_objects** command in both Legacy and Tcl format.

```
prompt> set_app_var query_objects_format Legacy
Legacy
prompt> query_objects [get_cells o*]
{or1 or2 or3}
prompt> query_objects -class cell U*
{U1 U2}
prompt> query_objects -verbose -class cell \
? [list U* [get_nets n1]]
{cell:U1 cell:U2 net:n1}
prompt> set_app_var query_objects_format Tcl
Tcl
prompt> query_objects [get_cells o*]
{or1 or2 or3}
prompt> query_objects -class cell U*
{U1 U2}
prompt> query_objects -verbose -class cell \
? [list U* [get_nets n1]]
{{cell U1} {cell U2} {net n1}}
```

When you omit the **-class** option, only those elements of the *object_spec* argument that are collections generate output. The other elements generate error messages.

```
prompt> query_objects [list U* [get_nets n1] n*]
```

```
Error: No such collection 'U*' (SEL-001)
Error: No such collection 'n*' (SEL-001)
{n1}
```

When the output is truncated, the command prints an the ellipsis at the end of the display. For the following example, assume the default truncation is 5 (it is actually 100).

```
prompt> query_objects [get_cells o*] -truncate 2
{"or1", "or2", ...}
prompt> query_objects [get_cells *]
{"or1", "or2", "or3", "U1", "U2", ...}
Output truncated (total objects 126)
```

SEE ALSO

```
collections(2)
get_cells(2)
get_clocks(2)
get_designs(2)
get_generated_clocks(2)
get_lib_cells(2)
get_lib_pins(2)
get_libs(2)
get_nets(2)
get_path_groups(2)
get_pins(2)
get_ports(2)
get_timing_paths(2)
```

query_port_net

Finds the net that is logically connected to a specified port. This is a UPF query command.

SYNTAX

```
string query_port_net  
    port_name  
    [-conn low | high]
```

Data Types

port_name string

ARGUMENTS

port_name

Specifies the name of the port to query for its connected net.

-conn low | high

Specifies whether to return the higher-level net (**high**, the default behavior), or lower-level net (**low**) connected to the port. For a top-level port, you must use **-conn low**, or else nothing is returned. For a port of a leaf-level cell, using this option is not allowed.

DESCRIPTION

The **query_port_net** command returns the net that is logically connected to the specified port. For a hierarchical port, the command returns the higher-level net by default. To return the lower-level net instead, use the **-conn low** option.

You can use this command at the tool shell prompt or in a script run by the **source** command, but not in a script run by the **load_upf** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the higher-level net connected to port I_LAT/Inst_1[30].

```
prompt> query_port_net I_LAT/Inst_1[30]  
I[30]
```

The following example reports the lower-level net connected to port I_LAT/Inst_1[30].

```
prompt> query_port_net I_LAT/Inst_1[30] -conn low  
I_LAT/Inst_1[30]
```

The following example demonstrates the reporting of a net connected to a top-level port, A_IN. Because the command reports the higher-level net by default, you must use the **-conn low** option to get the lower-level net name.

```
prompt> query_port_net A_IN  
prompt>  
prompt> query_port_net A_IN -conn low  
netA_IN
```

SEE ALSO

get_nets(2)
get_ports(2)
query_net_ports(2)

query_port_state

Returns information about the port states that have been previously defined for a specified supply port in the active scope. This is a UPF query command.

SYNTAX

```
list query_port_state
    port_name
    [-state state_name]
    [-detailed]
```

Data Types

```
port_name    string
state_name    string
```

ARGUMENTS

port_name

Specifies the supply port. The command reports the port states previously defined for this port.

-state state_name

Returns the **add_port_state** command that was used to create the specified state. If you omit this option, the command returns a list of all the defined states for the port.

-detailed

Causes the **-state** option to return the **add_port_state** command information in the form of a Tcl list of keyword and value pairs.

DESCRIPTION

The **query_port_state** command returns information about the port states that have been previously defined for a specified supply port. If you specify only the port name without any other options, the command returns a list of the defined port states. If you use the **-state state_name** option, the command returns the full **add_port_state** command previously used to create that state for the port. The **-detailed** option returns the same information in the form of a Tcl list of keyword and value pairs.

You can use this command at the tool shell prompt or in a script run by the **source** command, but not in a script run by the **load_upf** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples show how to report the port states defined for the supply port named "I_REG/I_REG_sw/out".

```
prompt> query_port_state I_REG/I_REG_sw/out  
{BLOCK BLOCK_off }
```

```
prompt> query_port_state I_REG/I_REG_sw/out -state BLOCK  
add_port_state I_REG/I_REG_sw/out -state {BLOCK {1.080000 }}
```

```
prompt> query_port_state I_REG/I_REG_sw/out -state BLOCK -detailed  
{port_name I_REG/I_REG_sw/out} {state_name BLOCK } {state {1.080000 }}
```

```
prompt> query_port_state I_REG/I_REG_sw/out -state *  
add_port_state I_REG/I_REG_sw/out -state {BLOCK {1.080000 }}  
add_port_state I_REG/I_REG_sw/out -state {BLOCK_off {off }}
```

SEE ALSO

add_port_state(2)
create_supply_port(2)

query_power_switch

Returns information about a power switch that was previously created in the active scope. This is a UPF query command.

SYNTAX

```
list query_power_switch
    switch_name
    [-detailed]
```

Data Types

switch_name string

ARGUMENTS

switch_name

Specifies the name of the power switch. The command returns the full **create_power_switch** command previously used to create the power switch. Use an asterisk (*) to get a list of all the power switch names.

-detailed

Returns the **create_power_switch** command information in the form of a Tcl list of keyword and value pairs.

DESCRIPTION

The **query_power_switch** command returns information about previously defined power switches. The command "**query_power_switch** *" returns a list of the power switch names. If you specify the name of the particular power switch, the command returns the full **create_power_switch** command previously used to create that power switch. The **-detailed** option returns the same information in the form of a Tcl list of keyword and value pairs.

You can use this command at the tool shell prompt or in a script run by the **source** command, but not in a script run by the **load_upf** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples show how to report the **add_pst_state** command previously used to create power state s2 in power state table PST1.

```
prompt> query_power_switch *  
{SW1 SW2 }
```

```
prompt> query_power_switch SW1  
create_power_switch SW1 -domain TOP -output_supply_port {out SN2}  
-input_supply_port {in SN1} -control_port {ctrl on} -on_state  
{state1 in {ctrl}}
```

```
prompt> query_power_switch SW1 -detailed  
{switch_name sw1} {domain TOP} {output_supply_port {out SN2}}  
{input_supply_port {in SN1}} {control_port {ctrl on}} {on_state  
{state1 in {ctrl}}}
```

SEE ALSO

```
create_power_switch(2)  
query_map_power_switch(2)
```

query_pst

Returns information about the power state tables that have been previously created in the active scope. This is a UPF query command.

SYNTAX

```
list query_pst
    table_name
    [-detailed]
```

Data Types

table_name string

ARGUMENTS

table_name

Specifies the name of the power state table. The command returns the full **create_pst** command previously used to create the power state table. Use an asterisk (*) to get a list of all power state table names.

-detailed

Returns the **create_pst** command information in the form of a Tcl list of keyword and value pairs.

DESCRIPTION

The **query_pst** command returns information about previously defined power state tables in the active scope. The command "**query_pst** *" returns a list of the power state table names. If you specify the name of the particular power state table, the command returns the full **create_pst** command previously used to create that power state table. The **-detailed** option returns the same information in the form of a Tcl list of keyword and value pairs.

You can use this command at the tool shell prompt or in a script run by the **source** command, but not in a script run by the **load_upf** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples show how to report the **create_pst** command previously used to create power state table PST1.

```
prompt> query_pst *  
{PST1 PST2 }
```

```
prompt> query_pst PST1  
create_pst PST1 -supplies {SP1 SP2 SP3 }
```

```
prompt> query_pst PST1 -detailed  
{table_name PST1} {supplies {SP1 SP2 SP3 }}
```

SEE ALSO

create_pst(2)
query_pst_state(2)

query_pst_state

Returns information about a state in a power state table that was previously created in the active scope. This is a UPF query command.

SYNTAX

```
list query_pst_state
    state_name
    -pst table_name
    [-detailed]
```

Data Types

```
state_name  string
table_name  string
```

ARGUMENTS

state_name

Specifies the name of the power state. The command returns the full **add_pst_state** command previously used to create the power state. Use an asterisk (*) to get a list of all the power states for the specified power state table.

-pst table_name

Specifies the name of the power state table. To get a list of the power state tables in the current scope, use the "**query_pst ***" command.

-detailed

Returns the **add_pst_state** command information in the form of a Tcl list of keyword and value pairs.

DESCRIPTION

The **query_pst_state** command returns information about the states that have been previously defined for a specified power state table. The command "**query_pst_state * -pst table_name**" returns a list of the power states defined for the specified table. If you specify the name of the particular power state, the command returns the full **add_pst_state** command previously used to create that power state. The **-detailed** option returns the same information in the form of a Tcl list of keyword and value pairs.

You can use this command at the tool shell prompt or in a script run by the **source** command, but not in a script run by the **load_upf** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples show how to report the **add_pst_state** command previously used to create power state s2 in power state table PST1.

```
prompt> query_pst *  
{PST1 PST2}
```

```
prompt> query_pst_state * -pst PST1  
{s1 s2 s3 }
```

```
prompt> query_pst_state s2 -pst PST1  
add_pst_state s2 -pst PST1 -state {s12 s11 s12 }
```

```
prompt> query_pst_state s2 PST1 -detailed  
{state_name s2} {pst PST1} {state {s12 s11 s12 }}
```

SEE ALSO

add_pst_state(2)
query_pst(2)

query_qor_snapshot

Analyzes timing report files from existing QoR snapshots, applies any specified filters, and displays the results in an appropriate format.

This command operates in two distinct modes: CTR (general timing) mode and CTS (clock tree synthesis) mode.

SYNTAX for CTR Mode

```
status query_qor_snapshot
  [-name snapshot_name]
  [-directory directory_name]
  [-display]
  [-type min | max]
  [-incremental]
  [-output_file file_name]
  [-output_false_paths file_name]
  [-sort_by column_list]
  [-group_by column_list]
  [-columns column_list]
  [-and column_list]
  [-filters filter_list]
  [-hierarchy]
  [-from module_list]
  [-to module_list]
  [-through module_list]
  [-output_group_paths file_name]
  [-group_path_prefix prefix]
  [-group_path_append appendix]
  [-case_sensitive]
  [-subgroup {[subgroup_name] [subgroup_options]}]
  [-infeasible_paths]
```

Data Types

```
snapshot_name    string
directory_name   string
file_name        string
column_list      list of strings
filter_list      string
module_list      list of strings
prefix           string
appendix         string
subgroup_options string
```

SYNTAX for CTS Mode

```
status query_qor_snapshot
  [-clock_tree_only]
  [-name snapshot_name]
  [-directory directory_name]
  [-display]
  [-cts_output_file file_name]
  [-cts_sort_by column_list]
  [-cts_columns column_list]
  [-cts_and column_list]
  [-cts_filters filter_list]
  [-cts_set_exceptions_file file_name]
  [-cts_clear_exceptions_file file_name]
  [-case_sensitive]
```

ARGUMENTS

-name *snapshot_name*

Specifies the name of the QoR snapshot to be analyzed.

This is a required option when you run this command the first time in the current session. In subsequent runs, if you do not specify this option, the report used in the last query is reused.

-directory *directory_name*

Specifies the name of the directory to search for the QoR snapshot specified by the **-name** option.

If you do not specify this option, the tool searches for the QoR snapshot in the directory specified by the **icc_snapshot_storage_location** variable.

-display

Displays the output using a built-in HTML viewer to allow for interactive viewing of selected paths in the layout window (Not supported for DCT). This option starts the GUI; do not use this option in batch mode. The **-display** option takes priority over (disables) the **-output_file** and **-cts_output_file** options.

-type min | max

Loads the appropriate snapshot setup report if you specify **max** and hold report if you specify **min**.

The default is **max** for loading setup.

-incremental

Queries the previous queried result instead of querying from scratch. This option is only valid when it is used with the **-hierarchy** option.

-output_file *file_name*

Specifies the name of the generated output files.

If you specify a file name that ends in .html, only an HTML output is created. If you specify a file name that ends in .txt, only a text output is created. Otherwise, both text and HTML outputs are created and saved into files with appropriate names. If the destination directory is read-only, an error message is displayed and you must specify a different file name.

-output_false_paths *file_name*

Generates a Tcl command file that contains the **set_false_path** commands. One **set_false_path** command is generated for each timing path that matches one or more filters. Only those attributes of the timing path that violate a filter are used to create each individual **set_false_path** command.

The set of **set_false_path** commands created is the same as the set that is created by selecting all paths and all columns in the HTML report or interactive report and then clicking the Create False Paths button.

After all **set_false_path** commands have been created, the tool creates a minimum set of commands to eliminate duplicates. The duplicate elimination process often causes the order of the **set_false_path** commands in the output file to be different than the order of the timing paths in the output file or input report.

The `_fp.tcl` suffix is appended to the specified file name.

-sort_by *column_list*

Specifies the column used to sort the results. The default is `wns`.

To sort a column in descending order, specify "*column_name*" or "*+column_name*". To sort a column in ascending order, specify "*-column_name*". Two exceptions: `wns` and `zero_path` columns have their default sorting order reversed (*+wns* results in an ascending sort). This is so that the default sort order for a column produces the most meaningful sorting order (descending for most numeric columns).

Valid sort columns are: `path_group`, `startpoint`, `endpoint`, `scenario`, `wns`, `transition_degradation`, `slew_degradation`, `zero_path`, `logic_levels`, `path_delay`, `input_delay`, `output_delay`, `clock_uncertainty`, `incremental`, `transition_time`, `capacitance`, `fanout`, `clock_skew`, `launch_clock`, `launch_clock_latency`, `capture_clock`, `capture_clock_latency`, `bufinvcnt`, `io_path`, and `cross_clock`.

-group_by *column_list*

Specifies the column used to group the results. This option groups the results based on the unique values found for this column to create a summary that links to details for that value of the column. Each detail page is sorted by the sort column.

Valid group columns are: `path_group`, `startpoint`, `endpoint`, `scenario`, `launch_clock`, `capture_clock`, `bufinvcnt`, `logic_levels`, and `cross_clock`.

-columns *column_list*

Specifies the list of columns to include in the output. The columns are displayed in the specified order. If you specify a column multiple times, its location is determined by the last instance. Use of the `instance` or `file_instance` filters as an output column is not recommended because the results are not meaningful.

The column names are specified in the following format:

column_name[*.width*]

where *column_name* is the same as the corresponding filter name (see the description for the **-filters** option for information about the filter names) and *width* is an optional integer value that defines the column width. Width is somewhat flexible in HTML output and the column will not expand past the width of the widest value. All available columns are printed by default.

Note: Not all the columns available in the IC Compiler tool are available in the Design Compiler product family tools. For example, if you specify the `bufinvcnt` column in Design Compiler, the tool does not recognize it and issues the following message:

No columns matched specification bufinvcnt

You can specify the column names as minimally-distinct strings. That is, instead of using `capacitance`, you can use `"capa"` or even

"paci" because no other column name contains that sequence of letters.

If you do not specify this option, the default columns are:

- Path_group, startpoint, endpoint, wns, zero_path, path_delay, input_delay, output_delay, clock_uncertainty, incremental, fanout, logic_levels, launch_clock, capture_clock.

-and column_list

Specifies a list of filters to apply with AND logic. These filters are applied to the paths that pass the filters specified in the **-filters** option. Paths that match one or more OR filters and all AND filters are included in the result.

For example, if you specify **-filters "-wns ,-3.0 -cross_clock 1"**, the command reports the paths with timing violations where WNS is worse than -3.0 OR -cross_clock is true. If you specify **-filters "-wns ,-3.0 -cross_clock 1" -and wns**, the command reports the paths with timing violations where WNS is worse than -3.0 and -cross_clock is true. If all specified filters are also specified as using **-and**, one will automatically be treated as OR.

-filters filter_list

Specifies a list of filters to apply to the paths analyzed from the timing report. The list must be enclosed in double quotes. The specified filters are applied with OR logic; all paths matching any specified filter are selected.

Each filter consists of a filter option and its argument. Most filters accept a range of either floating point or integer numbers as an argument. The argument range consists of two numbers separated by a comma. The first number is the lower bound and the second number is the upper bound. You can also specify just one number. In this case, it is interpreted as the lower bound, unless it is preceded by a comma. If the number is preceded by a comma, it is interpreted as the upper bound.

Filters can be grouped into two areas: default filters and other general filters. Default filters are used to indicate violations with respect to important matrices in the timing report, such as output pin transition degradation, bad net slew degradation, and long logic levels, that can be quantified. Other general filters exist to allow you to further narrow down the query searches.

You can specify one or more of the following filters:

- **-wns lower_bound,upper_bound**
Includes all paths that have a worst negative slack (WNS) within the specified floating point range.
- **-transition_degradation lower_bound,upper_bound**
Includes all paths that have their transition degradation ratio (output pin transition time / input pin transition time) within the specified floating point range.
- **-slew_degradation lower_bound,upper_bound**
Includes all paths that have a slew degradation (input pin transition / last stage output pin transition) within the specified floating point range.
- **-logic_level lower_bound,upper_bound**
Includes all paths that have their number of logic levels within the specified integer range.
- **-fanout lower_bound,upper_bound**
Includes all paths that have their worst fanout within the specified integer range.
- **-input_delay lower_bound,upper_bound**
Includes all paths that have an external input delay within the specified floating point range.
- **-output_delay lower_bound,upper_bound**
Includes all paths that have an external output delay within the specified floating point range.
- **-clock_uncertainty lower_bound,upper_bound**
Includes all paths that have clock uncertainty within the specified floating point range.
- **-zero_path lower_bound,upper_bound**
Includes all paths with a zero-path margin within the specified floating point range.

- `-clock_skew lower_bound,upper_bound`
Includes all paths that have clock skew within the specified floating point range.
- `-path_delay lower_bound,upper_bound`
Includes all paths that have a path delay within the specified floating point range.
- `-infeasible_paths value`
Includes all paths that have the specified value, YES or NO, under the Infeasible Paths column.
- `-transition_time lower_bound,upper_bound`
Includes all paths that have a transition time within the specified floating point range.
- `-capacitance lower_bound,upper_bound`
Includes all paths that have their worst capacitance within the specified floating point range.
- `-derate lower_bound,upper_bound`
Includes all paths that have their timing derate within the specified floating point range.
- `-incremental lower_bound,upper_bound`
Includes all paths that have an incremental value within the specified floating point range.
- `-bufinvcount lower_bound,upper_bound`
Includes all paths that have their buffer+inverter count within the specified integer range.
- `-io_path lower_bound,upper_bound`
Include all paths that contain an input or output element in their path. Input/Output elements are: (in), (inout) or (out).
- `-cross_clock lower_bound,upper_bound`
Includes all paths that have different values for the launch and capture clocks. Paths with the same launch and capture clocks will have a value of 0 for this attribute, and paths with different launch and capture clocks will have an attribute of 1.
- `-launch_clock pattern`
Includes all paths whose launch clock name matches the pattern. You can specify multiple patterns by using the OR operator, "|". Wildcards are supported in the pattern at the beginning or end.
- `-capture_clock pattern`
Includes all paths whose capture clock name matches the pattern. You can specify multiple patterns by using the OR operator, "|". Wildcards are supported in the pattern at the beginning or end.
- `-path_group pattern`
Includes all paths whose path group matches the pattern. You can specify multiple patterns by using the OR operator, "|". Wildcards are supported in the pattern at the beginning or end.
- `-startpoint pattern`
Includes all paths whose startpoint matches the pattern. You can specify multiple patterns by using the OR operator, "|". Wildcards are supported in the pattern at the beginning or end.
- `-endpoint pattern`
Includes all paths whose endpoint matches the pattern. You can specify multiple patterns by using the OR operator, "|". Wildcards are supported in the pattern at the beginning or end.
- `-scenario pattern`
Includes all paths whose scenario matches the pattern. You can specify multiple patterns by using the OR operator, "|". Wildcards are supported in the pattern at the beginning or end.
- `-instance pattern`
Includes all paths that contain an instance matching the pattern. You can specify multiple patterns by using the OR operator, "|". Wildcards are supported in the pattern.
- `-file_instance file_name`

Includes all paths that contain an instance that matches any of the patterns listed in the specified file (patterns can be separated by space or a newline). Patterns take the same form as those for `-instance`.

- `-libcell pattern`
Includes all paths that contain a library cell matching the pattern. You can specify multiple patterns by using the OR operator, "|". Wildcards are supported in the pattern.
- `-instance_mcnt lower_bound,upper_bound`
Includes all paths for which the total count of matches of any of `-instance` or `-file_instance` patterns is within the specified range.
- `-libcell_mcnt lower_bound,upper_bound`
Includes all paths for which the total count of matches of any `-libcell` pattern is within the specified range.
- `-tns lower_bound,upper_bound`
Includes all potential path groups with total negative slack (TNS) values within the specified floating point range.

You must use the **-hierarchy** option when you use this filter.

- `-nvp lower_bound,upper_bound`
Includes all potential path groups whose number of violating paths is within the specified integer range.

You must use the **-hierarchy** option when you use this filter.

Note: You can specify the filters as minimally-distinct strings. That is, instead of using capacitance, you can use "capa" or even "paci", since no other column name contains that sequence of letters.

If you do not specify this option, the following filters are applied:

```
-filters "-wns ,0 -zero_path ,0 -fanout 40 -transition_degradation 2.0
-slew_degradation 2.0 -logic_levels 50"
```

-hierarchy

Checks if timing violations are related to blocks, hard macros, or user-specified hierarchical modules (using the **-from**, **-through**, and **-to** options).

-from *module_list*

Specifies the instances to use as the from instance list. Each instance in this list is used as a startpoint for one or more potential path groups. Paths that start at one of these points might also pass through instances specified in the **-through** option and might end at instances specified in the **-to** option. Each combination is considered as a potential path group.

This option is valid only when you also use the **-hierarchy** option.

-to *module_list*

Specifies the instances to use as the to instance list. Each instance in this list is used as an endpoint for one or more potential path groups. Paths that end at one of these points might also start at instances specified in the **-from** option and might pass through instances specified in the **-through** option. Each combination is considered as a potential path group.

This option is valid only when you also use the **-hierarchy** option.

-through *module_list*

Specifies the instances to use as the through instance list. Each instance in this list is used as a through point for one or more potential path groups. Paths that pass through one of these points might also start at instances specified in the **-from** option and might end at instances specified in the **-to** option. Each combination is considered as a potential path group.

This option is valid only when you also use the **-hierarchy** option.

-output_group_paths *file_name*

Generates an SDC file containing the list of path groups identified by the **-hierarchy** option and other related options, such as **-filter**).

If you do not specify the **-output_group_paths** option, you must use the HTML browser to manually select the path groups to create from the HTML report. The `_gp.tcl` suffix is appended to the file name.

This option is valid only when you also use the **-hierarchy** option.

-group_path_prefix *prefix*

Specifies a string that is inserted at the beginning of any group names created by the Create Groups button in the top-level summary page displayed with the **-hierarchy** option of the **query_qor_snapshot** command. This option can be used only with the **-hierarchy** option. For example, **-group_path_prefix "CUSTOM"** will create path groups named CUSTOM_... .

-group_path_append *appendix*

Specifies a string that is appended to the end of any **group_paths** commands created by the Create Groups button in the top-level summary page displayed with the **-hierarchy** option of the **query_qor_snapshot** command. This option can be used only with the **-hierarchy** option. For example, **-group_path_append "-critical_range 0.5"** will cause any **group_path** commands created by the Create Groups button in the following format:

```
group_path ... -critical_range 0.5
```

-case_sensitive

Makes the option argument values and filters case sensitive.

-subgroup {*subgroup_name subgroup_options*}

Allows multiple queries with some restrictions in CTR mode. You can specify multiple **-subgroup** options in a single run of the **query_qor_snapshot** command.

The key difference between using the **-subgroup** option and running multiple queries on the same data set is that the set of timing paths in each subgroup is unique when using the **-subgroup** option. That is, the second subgroup uses those paths that do not meet the requirements of the first subgroup, the third subgroup uses those paths that do not meet the requirements of the first two subgroups, and so forth. After processing each specified subgroup, the tool creates a default subgroup named Others, which consists of all the remaining paths that are not included in the subgroups created by this option.

The results are summarized in a format similar to that used by the **-group_by** option; that is, one line per subgroup to show the paths found, worst negative slack (WNS) for the set of paths, and total negative slack (TNS) for the set of paths. You can click a subgroup name to see details about that subgroup.

If you do not specify the subgroup name, the tool assigns a name to the subgroup.

For each subgroup, you can specify zero or more of the following options to define the subgroup: **-columns**, **-case_sensitive**, **-filters**, **-and**, **-sort_by**, and **-group_by**. Any unspecified subgroup options have the same default values as a regular categorized timing report query.

To control the settings for the Others group, use the following command options (outside of the subgroup specification): **-columns**, **-sort_by**, and **-group_by**.

You can define a subgroup query with only **-subgroup { }**.

You cannot use this option with the **-hierarchy**, **-incremental**, and **-clock_tree_only** options or with clock tree reports.

-infeasible_paths

Enables the displaying of feasibility information of each path under the Infeasible Paths column. This information is collected during compile exploration. If the snapshot query is created without the **-infeasible_paths** option, the column is filled with N/A instead of YES or NO.

-clock_tree_only

Indicates the command should run in CTS (clock tree synthesis) mode instead of the default CTR (general timing) mode. All CTR arguments are ignored and CTS arguments become active.

-cts_output_file *file_name*

Specifies the name of the generated output files.

If you specify a file name that ends in .html, only an HTML output is created. If you specify a file name that ends in .txt, only a text output is created. Otherwise, both text and HTML outputs are created and saved into files with appropriate names. If the destination directory is read-only, an error message is displayed and you must specify a different file name.

-cts_sort_by *column_list*

Specifies the column used to sort the results in CTS mode. The default is skew.

To sort a column in descending order, specify "*column_name*" or "+*column_name*". To sort a column in ascending order, specify "-*column_name*". This is so that the default sort order for a column produces the most meaningful sorting order (descending for most numeric columns).

Valid CTS sort columns are: clock, scenario, master_clock, master_source, sinks_skew, path_delay, transition_violation, fanout_violation, capacitance_violation, skew_bottleneck_pins, skew_bottleneck_skew, skew_bottleneck_delay, explicit_ignore, dont_touch_subtree, dont_buffer_nets, dont_size_cells, hierarchy_preservation, size_only_cells, explicit_sync, explicit_nonstop, implicit_ignore, implicit_nonstop, etm_impossible_pins, etm_internal_pins, multi_output_skew_cells, multi_output_skew_diff, multi_timing_arc_cells, multi_timing_arc_diff, transition_degradation, slew_degradation, logic_levels and worst_incremental.

-cts_columns *column_list*

Specifies the list of columns to include in the output in CTS mode. The columns are displayed in the specified order. If you specify a column multiple times, its location is determined by the last instance.

The column names are specified in the following format:

column_name[.*width*]

where *column_name* is the same as the corresponding filter name (see the description for the **-cts_filters** option for information about the filter names) and *width* is an optional integer value that defines the column width. Width is somewhat flexible in HTML output and the column will not expand past the width of the widest value. To show all available columns in a predetermined order, specify **-cts_columns ALL**. The only column available for use with **-cts_columns** that is not described under the **-cts_filters** option is the "name" column, which shows the Pathname column (values are All, Longest, Shortest, Skew Bottleneck Pins).

Note: You can specify the column names as minimally distinct strings. That is, instead of using "capacitance_violation", you can use "capa" or even "e_viol", since no other column name contains that sequence of letters.

If you do not specify this option, the default is to display all available CTS columns.

-cts_and *column_list*

Specifies a list of filters to apply with AND logic. These filters are applied to the clocks that pass the filters specified in the **-cts_filters** option. Clocks that match one or more OR filters and all AND filters are included in the result.

For example, if you specify **-cts_filters {-skew ,-3.0 -etm_internal 1}**, the command reports the clocks which have a skew worse than -3.0 OR clocks which have one or more ETM internal pins. If you specify **-cts_filters {-skew ,-3.0 -etm_internal 1} -cts_and skew**, the command reports the paths with timing violations where skew is worse than -3.0 AND the clock has one or more ETM internal pins. If all specified filters are also specified with the **-cts_and** option, one of the filter is automatically treated as an OR filter.

-cts_filters *filter_list*

Specifies a list of filters to apply to the clocks analyzed from the clock tree report files. The list must be enclosed in double quotes. The specified filters are applied with OR logic; all clocks matching any specified filter are selected.

Each filter consists of a filter option and its argument. Most filters accept a range of either floating-point or integer numbers as the argument. The range argument consists of two numbers separated by a comma. The first number is the lower bound and the second number is the upper bound. You can also specify just one number. In this case, it is interpreted as the lower bound, unless it is preceded by a comma. If the number is preceded by a comma, it is interpreted as the upper bound.

Filters can be grouped into two areas: default filters and other general filters. Default filters are used to indicate violations with respect to important matrices in the clock tree report, such as output pin transition degradation, bad net slew degradation, and long logic levels, that can be quantified. Other general filters exist to allow you to further narrow down the query searches.

You can specify one or more of the following filters:

- `-skew lower_bound,upper_bound`
Include all clocks that have a skew within the specified floating point range.
- `-transition_degradation lower_bound,upper_bound`
Include all clocks that have their transition degradation ratio (output pin transition time divided by input pin transition time) within the specified floating-point range.
- `-slew_degradation lower_bound,upper_bound`
Include all clocks that have slew degradation (input pin transition divided by last stage output pin transition) within the specified floating-point range.
- `-logic_levels lower_bound,upper_bound`
Include all clocks that have their number of logic levels within the specified integer range.
- `-path_delay lower_bound,upper_bound`
Include all clocks that have the longest path delay within the specified floating-point range.
- `-sinks lower_bound,upper_bound`
Include all clocks having their number of sinks within the specified integer range.
- `-transition_violation lower_bound,upper_bound`
Include all clocks having their number of transition violations within the specified integer range.
- `-capacitance_violation lower_bound,upper_bound`
Include all clocks having their number of capacitance violations within the specified integer range.
- `-fanout_violation lower_bound,upper_bound`
Include all clocks having their number of fanout violations within the specified integer range.
- `-explicit_ignore lower_bound,upper_bound`
Include all clocks having their number of explicit ignore pins clock tree exceptions within the specified integer range.
- `-implicit_ignore lower_bound,upper_bound`
Include all clocks having their number of implicit ignore pins clock tree exceptions within the specified integer range.
- `-explicit_sync lower_bound,upper_bound`
Include all clocks having their number of explicit sync pins clock tree exceptions within the specified integer range.
- `-explicit_nonstop lower_bound,upper_bound`
Include all clocks having their number of explicit nonstop pins clock tree exceptions within the specified integer range.
- `-implicit_nonstop lower_bound,upper_bound`
Include all clocks having their number of implicit nonstop pins clock tree exceptions within the specified integer range.
- `-dont_touch_subtree lower_bound,upper_bound`
Include all clocks having their number of dont touch subtree pins clock tree exceptions within the specified integer range.

- `-dont_buffer_nets lower_bound,upper_bound`
Include all clocks having their number of dont buffer nets clock tree exceptions within the specified integer range.
- `-dont_size_cells lower_bound,upper_bound`
Include all clocks having their number of dont size cells clock tree exceptions within the specified integer range.
- `-hierarchy_preservation lower_bound,upper_bound`
Include all clocks having their number of hierarchy preservation objects clock tree exceptions within the specified integer range.
- `-size_only_cells lower_bound,upper_bound`
Include all clocks having their number of size only cells clock tree exceptions within the specified integer range.
- `-skew_bottleneck_pins lower_bound,upper_bound`
Include all clocks having their number of skew bottleneck pins within the specified integer range.

Skew bottleneck pins are driver pins in the clock tree having an increase in skew at the driver compared to the load pins. Skew at a driver is calculated, and then, skew at each load pin of the driver is calculated. The largest increase in skew from any load pin to the driver is then compared to the **-skew_bottleneck_threshold** value. If skew increase is greater than the threshold, the driver pin is marked as a skew bottleneck pin.

- `-skew_bottleneck_threshold lower_bound,upper_bound`
When calculating skew bottleneck pins, use this range to determine whether a pin is considered a skew bottleneck pin. The default value (even if not specified) is 0.05 (that is, lower bound of 0.05 or no upper bound).

Note: this filter has no corresponding column that can be shown with the **-cts_columns** option.

- `-skew_bottleneck_skew lower_bound,upper_bound`
Include all clocks having their worst skew bottleneck pin skew within the specified floating-point range.
- `-etm_impossible_pins lower_bound,upper_bound`
Include all clocks having their number of ETM impossible pins within the specified integer range.
- `-etm_impossible_threshold lower_bound,upper_bound`
When marking ETM impossible pins, use this range to determine whether a pin is considered an ETM impossible pin. The default value (even if not specified) is 0.01 (that is, lower bound of 0.01 and no upper bound).

Note: this filter has no corresponding column that can be shown with the **-cts_columns** option.

- `-etm_internal lower_bound,upper_bound`
Include all clocks having their number of ETM internal pins within the specified integer range.
- `-multi_output_skew_cells lower_bound,upper_bound`
Include all clocks having cells with multiple outputs and skew at these outputs within the specified integer range.
- `-multi_output_skew_threshold lower_bound,upper_bound`
When marking multiple output skew pins, use this range to determine whether a pin is considered a multiple output skew pin. The default value (even if not specified) is 0.05 (that is, lower bound of 0.05 and no upper bound).

Note: this filter has no corresponding column that can be shown with the **-cts_columns** option.

- `-multi_output_skew_diff lower_bound,upper_bound`
Include all clocks having cells with multiple outputs and the worst skew at the outputs within the specified floating-point range.
- `-multi_timing_arc_cells lower_bound,upper_bound`
Include all clocks having cells with multiple timing arcs and the number of arcs within the specified integer range.
- `-multi_timing_arc_threshold lower_bound,upper_bound`
When marking multiple timing arc pins, use this range to determine whether a pin is considered a multiple timing arc pin. The default value (even if not specified) is 0.05 (that is, lower bound of 0.05 and no upper bound).

Note: this filter has no corresponding column that can be shown with the **-cts_columns** option.

- **-multi_timing_arc_diff *lower_bound,upper_bound***
Include all clocks having cells with multiple timing arcs and the worst difference of the timing arcs within the specified floating-point range.
- **-worst_incremental *lower_bound,upper_bound***
Include all clocks that have a cell with incremental cell value within the specified floating-point range.
- **-scenario *pattern***
Include all clocks whose scenario name matches any of the specified patterns. You can specify multiple patterns as a space-separated list. The specified patterns are applied with OR logic; all scenarios matching any specified pattern are selected. Wildcards are supported in patterns at the beginning or at the end.
- **-clock *pattern***
Include all clocks whose clock name matches any of the specified patterns. You can specify multiple patterns as a space-separated list. The specified patterns are applied with OR logic; all clock names matching any specified pattern are selected. Wildcards are supported in patterns at the beginning or at the end.
- **-master_clock *pattern***
Include all clocks whose master clock name matches any of the specified patterns. You can specify multiple patterns as a space-separated list. The specified patterns are applied with OR logic; all master clock names matching any specified pattern are selected. Wildcards are supported in patterns at the beginning or at the end.
- **-master_source *pattern***
Include all clocks whose master source name matches any of the specified patterns. You can specify multiple patterns as a space-separated list. The specified patterns are applied with OR logic; all master source names matching any specified pattern are selected. Wildcards are supported in patterns at the beginning or at the end.

If you do not specify this option, the following filters are applied:

```
-cts_filters {-transition_violation 1 -fanout_violation 1
-capacitance_violation 1 -dont_touch 1 -dont_buffer 1 -dont_size 1
-size_only 1 -explicit_ignore 1 -skew 0 -transition_deg 2.0 -slew_deg 2.0}
```

-cts_set_exceptions_file *file_name*

Generates a Tcl command file containing **set_clock_tree_exceptions** commands. One or more **set_clock_tree_exceptions** commands are generated for each clock path matching one or more filters that can create exceptions. The filters that can create clock tree exceptions are extracted timing model (ETM) internal pins, ETM impossible pins, multiple timing arc cells, and multiple output skew cells. If none of these filters are specified, this output file will be empty.

The set of **set_clock_tree_exceptions** commands created is the same as selecting all paths and all columns that can create clock tree exceptions in the HTML report or interactive report and then selecting the **Remove/Create Exceptions** button.

After creating all **set_clock_tree_exceptions** commands, a minimum set of commands is created by eliminating duplicates. The duplicate elimination process often causes the order of the **set_clock_tree_exceptions** commands in the output file to be different than the order of the clock paths in the output file or input report.

The **_se.tcl** suffix is appended to the specified file name.

-cts_clear_exceptions_file *file_name*

Generates a Tcl command file that contains **remove_clock_tree_exceptions** commands. One or more **set_clock_tree_exceptions** commands are generated for each clock path that matches one or more filters that can remove exceptions. The filters that can generate remove clock tree exception commands are don't touch subtree, don't buffer nets, don't size cells, size-only cells, explicit sink pins, explicit nonstop pins, and hierarchy preservation. If none of these filters are specified, this output file will be empty.

The set of **remove_clock_tree_exceptions** commands created is the same as selecting all paths and all columns that can generate remove clock tree exception commands in the HTML report or interactive report, and then selecting the **Remove/Create Exceptions** button.

After creating all **remove_clock_tree_exceptions** commands, a minimum set of commands is created by eliminating duplicates. The duplicate elimination process often causes the order of the **remove_clock_tree_exceptions** commands in the output file to be different than the order of the clock paths in the output file or input report.

The `_ce.tcl` suffix is appended to the specified file name.

DESCRIPTION

The **query_qor_snapshot** command analyzes timing report files from existing QoR snapshots, applies any specified filters, and displays the results in an appropriate format.

The **query_qor_snapshot** command operates in two distinct modes: CTR (general timing) mode and CTS (clock tree synthesis) mode. When you use the **-clock_tree_only** option, the command operates in CTS mode and all CTR options are ignored. Similarly, in the default CTR mode, all CTS options are ignored. General options are applicable to both modes.

CTR Mode

In CTR mode, this command reads in a timing report file created by the **create_qor_snapshot** command and analyzes it. It shows a summary of the timing report, subject to one or more filters specified. All values that match a filter are highlighted with a * for text or red text for HTML. You can specify which columns to output with the **-columns** option, sorting with the **-sort_by** option, and which filters to apply as AND filters with the **-and** option.

The **query_qor_snapshot-hierarchy** command reads in the same timing report file and attempts to group violated paths across hierarchical modules, based on criteria you provide. The potential path groups are then listed based on the combinations of modules specified by the **-from**, **-to**, and **-through** options. If you do not specify hierarchical modules by using the **-from**, **-through**, and **-to** options, the **query_qor_snapshot-hierarchy** command automatically finds all the blocks and hard macros and looks for timing violation paths that cross all blocks and hard macros.

CTS Mode

In CTS mode, this command reads in a group of clock report files created by the **create_qor_snapshot -clock_tree_only** command and analyzes them.

The command reports a summary of the clock tree, subject to one or more specified filters. All values that match a filter are highlighted (with a * for text or red text for HTML). You can specify columns to display with the **-cts_columns** option, column value sorting order with the **-cts_sort_by** option, and filters to apply as AND filters with the **-cts_and** option.

Strings and Collections

Many commands, such as the **get_cells** command, return collections of objects. The **query_qor_snapshot** command is intended to run in a very minimalist environment without requiring a lot of design information to be loaded. As a result, collections are not supported as inputs to the **query_qor_snapshot** command. All input must be in the form of text strings. You can use the **join** and **collection_to_list** commands to turn a collection into a list of strings. See the CTR EXAMPLES section for an example.

CTR EXAMPLES

Before running the **query_qor_snapshot** command, you must run the **create_qor_snapshot** command to generate the snapshot data, as shown in the following example:

```
prompt> create_qor_snapshot -name place_opt
```

```
prompt> create_qor_snapshot -name place_opt_zwl -zero_wire_load
```

In the following example, the **query_qor_snapshot** command analyzes a QoR snapshot named preroute, which is saved under the snapshot directory and reports the values with WNS between -2.0 and -1.0.

```
prompt> query_qor_snapshot -name preroute -filters "-wns -2.0,-1.0"
```

In the following example, the **query_qor_snapshot** command analyzes QoR snapshot named placeopt, which is saved under the snapshot directory and reports the values with WNS less than -1.0 or fanout over 40.

```
prompt> query_qor_snapshot -name placeopt -filters "-wns ,-1.0 -fanout 40"
```

In following example, the **query_qor_snapshot** command reports the groups with violated timing paths through all blocks and hard macros.

```
prompt> query_qor_snapshot -hierarchy
```

In the following example, the **query_qor_snapshot** command reports the groups with violated timing path (WMS < 0) between the modules specified in the **-from** and **-to** options.

```
prompt> set user_list [[join [collection_to_list \
[get_cells {A/B C/D E F} ]]]
prompt> query_qor_snapshot -hierarchy -from $user_list -to $user_list
```

In the following example, the **query_qor_snapshot** command reports the groups with TNS more than -100 ns or timing violations with WNS worse than -3 ns. If the worst violation is -3.1 ns, this group is reported. If the worst slack is -2.9 ns, the command checks the TNS of these violations. If the TNS is more than 100 ns, these groups are reported; otherwise, these groups are not reported.

```
prompt> set user_list [collection_to_list [get_cells {A/B C/D E F} ]]
prompt> query_qor_snapshot -hierarchy -from $user_list -to $user_list \
-filter "-wns ,-3.0 -tns ,-100"
```

The following example shows all paths that have WNS less than zero.

```
prompt> query_qor_snapshot -name timing_report -filters "-wns ,0"
```

The following example shows all paths that have cells with output or input transition time over 2 ns.

```
prompt> query_qor_snapshot -name timing_report \
-filters "-transition_degradation 2"
```

The following example shows all paths that have path stages with slew degradation over 3 ns.

```
prompt> query_qor_snapshot -name timing_report \
-filters "-slew_degradation 3"
```

The following example shows all paths that have over 50 logic levels:

```
prompt> query_qor_snapshot -name timing_report \
-filters "-logic_level 50"
```

The following example shows all paths that have cells with a fanout over 40.

```
prompt> query_qor_snapshot -name timing_report -filters "-fanout 40"
```

The following example looks at all timing paths that match the worst negative slack (WNS) filter or zero-path filter. If the WNS violation is more than -10 ns or the path violates zero-path delay, the command automatically generates appropriate **set_false_path** commands for those timing paths.

```
prompt> query_qor_snapshot -name timing_report \
-filter "-wns ,-10.0 -zero_path ,0" \
-output_false_paths "my_"
```

The following example looks at all timing paths that match the WNS or zero_path filters. If the slack violation is more than -10 ns or the path violates zero-path delay, the command automatically generates **group_path** commands for the modules specified in the **-from**, **-through**, and **-to** options.

```
prompt> query_qor_snapshot -name timing_report -hierarchy \
  -filter "-wns ,-10.0 -zero_path ,0" \
  -output_group_paths "my_sdc"
```

The following example defines a subgroup query that creates three subgroups. The subgroups are named Group 1, Query 2, and Others. Subgroup Group 1 contains paths with WNS less than -10.0 or zero-path margin less than 0. Subgroup Query 2 contains paths not matched by subgroup Group 1 that also match the default filters. The Others subgroup contains any remaining paths that are not matched by the first two subgroups.

```
prompt> query_qor_snapshot -name timing_report \
  -subgroup { "Group 1" -filter "-wns ,-10.0 -zero_path ,0" } \
  -subgroup { }
```

CTS EXAMPLES

Before running the **query_qor_snapshot** command in CTS mode, you must run the **create_qor_snapshot** command to generate the snapshot data, as shown in the following example:

```
prompt> create_qor_snapshot -name clock_opt -clock_tree
```

The following example looks at all clock paths that match the ETM impossible pins filter. If the clock path has one or more ETM impossible pins, the command automatically generates appropriate **set_clock_tree_exceptions** commands for those clock paths.

```
prompt> query_qor_snapshot -clock_tree_only -name clk_report \
  -cts_filters {-etm_impossible_pins 1} \
  -cts_set_exceptions_file {my_}
```

The following example looks at all clock paths that match the don't touch subtree filter. If the clock path has one or more don't touch subtree exceptions, the command automatically generates appropriate **remove_clock_tree_exceptions** commands for those clock paths.

```
prompt> query_qor_snapshot -clock_tree_only -name clk_report \
  -cts_filters {-dont_touch_subtree 1} \
  -cts_clear_exceptions_file {my_}
```

In the following example, the **query_qor_snapshot** command reports all clocks with skew greater than 5 ns.

```
prompt> query_qor_snapshot -clock_tree_only -name clk_report \
  -cts_filters {-skew 5.0}
```

In the following example, the **query_qor_snapshot** command reports all clocks that have cells with input or output transition time over 2 ns:

```
prompt> query_qor_snapshot -clock_tree_only -name clk_report \
  -cts_filters {-transition_degradation 2}
```

In the following example, the **query_qor_snapshot** command reports all clocks that have ETM impossible pins and sets the ETM impossible pin threshold to 0.1.

```
prompt> query_qor_snapshot -clock_tree_only -name clk_report \
  -cts_filters {-etm_impossible_pins 1 -etm_impossible_threshold 0.1}
```

In the following example, the **query_qor_snapshot** command reports all clocks that have path segments with slew degradation over 3 ns:


```
prompt> query_qor_snapshot -clock_tree_only -name clk_report \  
-cts_filters {-slew_degradation 3}
```

In the following example, the **query_qor_snapshot** command reports all clocks that have over 50 logic levels:

```
prompt> query_qor_snapshot -clock_tree_only -name clk_report \  
-cts_filters {-logic_level 50}
```

SEE ALSO

[create_qor_snapshot\(2\)](#)
[gui_online_browser\(3\)](#)

quit!

quits the application without posting an application exit dialog

SYNTAX

quit!

ARGUMENTS

None.

DESCRIPTION

This command is an alternative to the **quit** command. When called with the GUI up, the **quit** command posts an application exit dialog. To avoid having to "deal with" the exit dialog, the user can instead call the **quit!** command which is guaranteed to exit the application without posting a GUI exit dialog.

EXAMPLES

prompt> **quit!**

SEE ALSO

quit(2)

quit

Exits the shell.

SYNTAX

string **quit**

ARGUMENTS

The **quit** command has no arguments.

DESCRIPTION

The **quit** command exits from the application. It is basically a synonym for the **exit** command with no arguments.

EXAMPLES

The following example exits the current session:

```
prompt> quit
```

SEE ALSO

[exit\(2\)](#)

read

Reads from a channel.

SYNTAX

read

ARGUMENTS

The **read** command has no arguments.

DESCRIPTION

The **read** command reads data from the specified channel.

If the specified channel is in nonblocking mode, the command might not read as many bytes as requested; once all available input has been read, the command returns the available data rather than blocking for more input.

The **read** command translates end-of-line sequences in the input into newline characters according to the **-translation** option for the channel. For details, see the man page for the **fconfigure** command.

SEE ALSO

eof(n)
fblocked(n)
fconfigure(n)

read_bsdI

Reads the boundary-scan description language (BSDI) file for a boundary-scan design.

SYNTAX

```
status read_bsdI  
[-add_linkage_as_design_port true | false]  
file_name
```

Data Types

file_name string

ARGUMENTS

-add_linkage_as_design_port true | false

Enables or disables the addition of linkage ports specified in the input BSDI file in BSD patterns.

This option has no effect when the the design netlist is already available prior to this command.

When the option is set to **true** and the design netlist is not available prior to this command, the linkage ports specified in the BSDI file are treated as design ports in BSD patterns.

When the option is set to **false** and the design netlist is not available prior to this command, the linkage ports specified in the BSDI file are not treated as design ports in BSD patterns.

The default value of this option is **false**, that is, linkage ports of BSDI files are not used in BSD patterns.

file_name

Specifies the name of the input BSDI file.

DESCRIPTION

The **read_bsdI** command reads the BSDI file for a boundary-scan design and generates an internal data structure for use with BSD pattern generation by the **create_bsd_patterns** command.

Before reading a BSDI file, you can optionally read in a netlist for the top-level design. This netlist can be the full netlist or a simplified representation containing using black boxes. If you use a black-box representation, the design must contain at least the I/O pads. After reading in the netlist, set the current design to the top level with the **current_design** command.

You must always specify the target and link libraries before running the **read_bsdI** command, even if you are not reading in a netlist representation. Otherwise, this command fails due to being unable to read the link libraries.

EXAMPLES

The following example reads a design netlist and BSDL file for a boundary-scan design, then generates patterns into various formats for the boundary-scan logic of the design:

```
prompt> set search_path ". /lib $search_path"
prompt> set synthetic_library {standard.sldb dw_foundation.sldb}
prompt> set target_library {class.db}
prompt> set link_library [concat "*" $target_library $synthetic_library]
prompt> read_file -format verilog chip_bsd.v # optional
prompt> current_design test # optional
prompt> link # optional
prompt> read_bsdI chip_bsd.bsdI
prompt> create_bsd_patterns
prompt> write_test -f stil_testbench -o design_stil_tb
prompt> write_test -f verilog -o design_verilog_tb
prompt> write_test -f wgl_serial -o design_wgl_tb
```

SEE ALSO

create_bsd_patterns(2)
current_design(2)
read_file(2)
write_test(2)

read_cell_expansion

Reads the cell expansion data, including the area, width, and height of each cell that is expanded.

SYNTAX

```
int read_cell_expansion
    input_file_name
    [-reset]
```

Data Types

input_file_name string

ARGUMENTS

input_file_name

Specifies the name of the input expansion data file. This is a required argument.

-reset

Resets all cell expansion cells.

DESCRIPTION

During congestion optimization, cells in congested regions are expanded to occupy more space to create lower density areas. The **read_cell_expansion** command reads the cell expansion data, including the area, width, and height of each cell that is expanded. The data can be used for better congestion correlation between the Design Compiler Graphical tool and the IC Compiler and IC Compiler II tools.

EXAMPLES

The following example reads a cell expansion file and applies it to the current design.

```
prompt> read_cell_expansion in.exp
```

SEE ALSO

`write_cell_expansion(2)`

read_db

Reads in one or more design or library files in Synopsys database (.db) format.

SYNTAX

```
string read_db  
    file_names
```

Data Types

file_names list

ARGUMENTS

file_names

Specifies names of one or more files to be read.

DESCRIPTION

The **read_db** command reads design information from Synopsys database (.db) files into dc_shell.

This command is derived from the **read_file** command with the **-format db** option. For more information, see the man page for the **read_file** command.

SEE ALSO

read_file(2)

read_ddc

Reads in one or more design files in .ddc (Synopsys logical database) format.

SYNTAX

```
status read_ddc
    file_names
    [-scenarios scenario_list]
    [-active_scenarios active_scenario_list]
```

Data Types

<i>file_names</i>	list
<i>scenario_list</i>	list
<i>active_scenario_list</i>	list

ARGUMENTS

file_names

Specifies the names of one or more files to be read. If specifying more than one file name, enclose the names in braces ({}).

-scenarios *scenario_list*

Specifies the scenarios whose constraints are to be read from the .ddc file.

Constraints for any scenarios that are not listed are not read, which might save memory and runtime if the file is large and certain scenarios are not required in the current session.

If you do not use this option, all scenarios are read from the file. If you use this option with an empty *scenario_list* argument, no scenarios are read.

-active_scenarios *active_scenario_list*

Specifies a list of scenarios to be made active as the file is read. Any scenarios that are not listed are restored in the inactive state.

You can use this option only when you also specify the **-scenarios** option. The scenarios specified in the **-active_scenarios** option must be in the list of scenarios specified in the **-scenarios** option.

If you do not use this option, the active or inactive state of the scenarios are restored according to the state that was recorded when the file was written (unless the scenarios already exist).

DESCRIPTION

The **read_ddc** command reads design information from the Synopsys logical database (.ddc) files into memory.

This command is derived from the **read_file** command with the **-format ddc** option. For more information, see the man page for the **read_file** command.

Multicorner-Multimode Support

This command uses information from both active and inactive scenarios.

SEE ALSO

`read_file(2)`

read_file

Reads designs or libraries into memory, or reads libraries into the shell.

SYNTAX for dc shell

```
list read_file
  file_list
  [-define macro_names]
  [-format format_name]
  [-library library_name]
  [-rtl]
  [-single_file single_file_name]
  [-work library_name]
  [-scenarios scenario_list
    [-active_scenarios active_scenario_list]]
  [-autoread
    -top top_design_name
    [-recursive]
    [-exclude exclude_list]
    [-param param_list]
    [-output_script file_name]
    [-verbose]
    [-rebuild]]
```

Data Types

<i>file_list</i>	list
<i>macro_names</i>	list
<i>format_name</i>	string
<i>library_name</i>	string
<i>single_file_name</i>	string
<i>scenario_list</i>	list
<i>active_scenario_list</i>	list
<i>top_design_name</i>	string
<i>exclude_list</i>	list
<i>param_list</i>	list
<i>file_name</i>	string

SYNTAX for lc shell

```
list read_file
  file_list
```

Data Types

file_list list

ARGUMENTS

file_list

Specifies a list of files or directories to read. When specifying more than one file or directory, separate the names with a space and enclose the list of names in braces ({}).

-define *macro_names*

Specifies a list of top-level macros. This option can be used only with the Verilog and SystemVerilog formats.

-format *format_name*

Specifies the format in which a design is read. The *format_name* can be one of the following formats:

Format	Description
ddc	Synopsys internal database format (default)
db	Synopsys internal database format (libraries only)
verilog	IEEE Standard Verilog
sverilog	IEEE Standard SystemVerilog
vhdl	IEEE Standard VHDL
equation	Synopsys equation format
pla	Berkeley (Espresso) PLA format
st	Synopsys State Table format

If the **-autoread** option is used, only verilog, sverilog, and vhdl formats are allowed. Using both **-format** and **-autoread** makes the autoread flow look for files in the given language that match the extensions in the corresponding extensions list by the **hdlin_autoread_verilog_extensions**, **hdlin_autoread_sverilog_extensions**, or **hdlin_autoread_vhdl_extensions** variable. By default, the **-autoread** option reads in all files it finds with Verilog, SystemVerilog, and VHDL extensions, so specifying the **-format** option is optional.

-library *library_name*

Remaps the work library to *library_name*. This option can be used only with the VHDL format.

By default, the **analyze** command stores all output in the work library. To store design elements in libraries other than library specified by using the **-work** option, use the **-library** option.

-rtl

Specifies that an RTL design is being read. This option can be used only with the Verilog or VHDL format.

When used, the **read_file** command invokes HDL Compiler directly without attempting to determine if the specified files are gate-level netlists.

You cannot use this option with the **-autoread** option.

-single_file *single_file_name*

Puts the designs in the *file_list* argument into the specified internal design file, regardless of the input format.

You cannot use this option with the **-autoread** option.

-work *library_name*

Remaps the work library in the same way as the **-library** option. It is an alias for the **-library** option for VHDL only.

You cannot use this option with the **-autoread** option.

-scenarios *scenario_list*

Specifies the scenarios whose constraints are to be read from the .ddc file. This option can be used only with the ddc format.

Constraints for any scenarios that are not listed are not read, which might save memory and runtime if the file is large and certain scenarios are not required in the current session.

If you do not use this option, all scenarios are read from the file. If you use this option with an empty *scenario_list* argument, no scenarios are read.

You cannot use this option with the **-autoread** option.

-active_scenarios *active_scenario_list*

Specifies a list of active scenarios when the file is read. Any scenarios that are not listed are restored in the inactive state. This option can be used only with the ddc format.

You can use this option only when you also specify the **-scenarios** option. The scenarios specified in the **-active_scenarios** option must be in the list of scenarios specified in the **-scenarios** option.

If you do not use this option, the active or inactive state of the scenarios are restored according to the state that was recorded when the file was written (unless the scenarios already exist).

You cannot use this option with the **-autoread** option.

-autoread

Enables the **-autoread** option for the **read_file** command, which is supported only for the Verilog, SystemVerilog, and VHDL formats.

When the **-autoread** option is used, the RTL source files listed in **list_files** (or reachable using the **-recursive** option) are read in, their dependencies are inferred, the files are ordered for a proper analyze phase regarding their dependencies, the files are analyzed (internally using the **analyze** command), and then the top design defined by **-top** is elaborated in top-down mode.

-top *top_design_name*

Specifies the name of the top-level design. Elaboration starts at the specified design and elaborates the entire design hierarchy under the top design using the elaboration parameters defined by the optional **-param** option.

The current design is set to the design defined as top.

Any source file provided in *file_list* that is not required for top-down elaboration of the defined top module hierarchy is not analyzed.

This option is required when you specify the **-autoread** option. It is not valid without the **-autoread** option.

-recursive

Specifies that the command looks into the subdirectories of the directories specified by the *file_list* argument and recursively collects source files from those locations.

By default, the command looks into the specified directories, but it does not look in their subdirectories.

This option is available only with the **-autoread** option.

-exclude *exclude_list*

Specifies a list of files and directories that are not to be analyzed. If the command expands an item from the *file_list* argument, it

checks the file or directory against all entries in this list and ignores the file or directory if it is covered by at least one of the entries.

This option is available only with the **-autoread** option.

-param *param_list*

Specifies a list of design parameters enclosed in quotes for the design's elaboration.

Parameters within the list must be separated by commas. A specification can be based on parameter order (for example, "8,7,5") or on parameter names (for example, "N=>8,M=>6"). You can mix ordered and named parameter specifications as long as the ordered parameters are listed first.

The full *param_list* syntax is summarized in the DESCRIPTION section of the **elaborate** command man page.

This option is available only with the **-autoread** option.

-output_script *file_name*

Creates a Tcl script that is compatible with Design Compiler and Formality. The file reading order for third-party tools and a DOT language file-dependency graph are embedded in the script as comments.

This option is available only with the **-autoread** option.

-verbose

Prints out more messages for the **-autoread** option.

This option is available only with the **-autoread** option.

-rebuild

Allows you to start a clean run with the **-autoread** option that analyzes all required files whether they were analyzed before.

The **-autoread** option runs by default in update mode. This means that if a file is changed, the file and all dependent files are reanalyzed.

This option is available only with the **-autoread** option.

DESCRIPTION

This command reads the designs specified in the *file_list* argument into memory.

The current design is automatically set to the first design that is read, unless the **-autoread** option is used and the *current_design* will be the design specified with the **-top** option.

The **read_file** command can also read the libraries in *file_list* into the shell. Libraries are the only .db files that can be read; user designs can no longer be read from .db files.

While it is good practice to always specify the **-format** option, if you do not specify this option, the tool attempts to infer the format based on the file name extensions, if any. The recognized extensions are .ddc for ddc format; .db, .pdb, .sdb, or .sldb for db format; .v or .verilog for Verilog format; .sv or .sverilog for SystemVerilog format; and .vhd or .vhdl for VHDL format. Extensions are not case-sensitive. If the file name ends in .gz, the preceding extension, if any, is used. For example, an extension of .v.gz is recognized as Verilog format. (The .gz extension is not supported for .ddc files.) If you specify multiple files on the same command line, their formats must all be the same. If the format cannot be inferred from the file names, the default format, ddc, is used.

Internally, designs are stored in design files. An internal design file is a "container" of designs and has a complete, unique, UNIX-type file name associated with it. The designs in a design file also have unique names, but designs with the same name can exist in

different design files. For information about displaying design files and their contents, see the **list_designs** man page.

By default, when the tool reads files in a format other than ddc or db, it places each design in its own internal design file that uses the following naming convention: *input_file_path/design_name.ddc*. The names of files read in db or ddc format remain the same.

When the tool reads an HDL parameterized design file, the design is not converted to a .ddc file, but instead is parsed and stored in an intermediate format. Then, you can build the design by using the **elaborate** command or by instantiating the design in another HDL design.

Because internal database files at lower levels of hierarchy are automatically read when they are linked, you do not need to read them separately. Designs are not automatically linked at the time they are read. You can use the **link** command after the **read_file** command to ensure that referenced designs are read correctly. For information on automatic loading, see the **link** man page.

An input file can have either a simple or a complex file name. A simple file name has no directory specification, which means (in UNIX) that it does not contain a slash (/). Simple file names, such as *adder.ddc* or *library.db*, must be found in a directory listed in the **search_path** variable. A complex file name has a directory specification in it, which means that it does contain one or more slashes. Complex file names, such as *./test.ddc*, *../test.ddc*, or *~/john/test/test.ddc*, are not searched for in the **search_path** variable.

Internally, libraries are stored in library files. An internal library file is a "container" of libraries and has a complete, unique, UNIX-type file name associated with it. The libraries in a library file also have unique names, but libraries with the same name can exist in different library files. For information about displaying library files and their contents, see the **list_libs** man page.

By default, when the tool reads non-database format files, each library is placed in its own internal library file that uses the following naming convention: *input_file_path/file_name.db:library_name*. The names of files read in db format remain the same.

Multicorner-Multimode Support

By default, this command uses information from both active and inactive scenarios. You can select different scenarios by using the **-scenarios** option.

-autoread Option Support

The **-autoread** option reads in the source files of a design listed in the *file_list*, determines the dependencies between them, orders the files regarding its dependencies, analyzes them in the correct order to avoid errors due missed or misplaced files and elaborates the design starting at the specified top-level module. It retains the resulting GTECH representation in memory.

The dependencies are calculated only from the files or directories present in *file_list*. If *file_list* changes between consecutive calls with the **-autoread** option, the dependency inference is performed over the latest set of files provided. Based on this, all required sources must be present in *file_list* option (or reachable if the **-recursive** option is used) on each call with the **-autoread** option.

When the **-autoread** option is enabled, the **read_file** command reads in all files listed in *file_list* with Verilog, SystemVerilog, and VHDL extensions, but only files required for the top design elaboration will be analyzed. For analyzing all sources, see the **analyze** command man page with the **-autoread** option support.

You can specify the parameters required for elaborating the top design by using the **-param** option. Its syntax and usage is the same as described for the **-parameters** option of the **elaborate** command. See the **elaborate** man page for more information.

To locate the source files, the **-autoread** option uses the **search_path** variable to expand each item listed by the *file_list* command argument. It then determines whether the result is excluded by the **-exclude** option or the **hdlin_autoread_exclude_extensions** variable. If it is not excluded the language of the sources is inferred from the file extensions that match with the corresponding extensions list set by the **hdlin_autoread_verilog_extensions**, **hdlin_autoread_sverilog_extensions**, or **hdlin_autoread_vhdl_extensions** variable for Verilog, SystemVerilog, or VHDL language respectively.

You can also use the **-autoread** option when you want to specify explicit file names in the *file_list* argument that do not have one of the language-specific extensions, but you have explicit say the language using the **-format** option.

When expanding a directory, the command collects the files from the directory and recursively from its subdirectories if the -

recursive option is set. The command then performs all extension checks on these files. If the **-format** option is set, only files with Verilog, SystemVerilog, or VHDL extensions are collected, based on the value of the option.

After the **-autoread** option collects all of the source files, it performs the following dependency checks:

- Detects analyze dependencies: Order RTL files for analyzing them the right order. For example, analyze the file that contains a VHDL entity before analyzing files that define architectures of that entity, or analyzes the file that contains a SV package declaration before the files that imports that package.
- Detects Verilog and SystemVerilog compilation unit dependencies: determines if a file needs to be analyzed in the same compilation unit with other files. For example, if a file defines macros, SV local parameter, SV enumerated values and the file is not explicitly included by the file that requires that definitions, then **-autoread** option and groups them in the same compilation unit in the correct order. This might not always be possible, such as when a macro is defined several times in different files and the **-autoread** cannot determine which of those alternatives is the right choice.
- Detects link dependencies: Schedule analyze stage for files required for elaboration of the design hierarchy. For example, if a (S)Verilog design that is instantiated in one source file is declared on a different source file provided in the *file_list*, the second file also requires to be analyzed for a complete top-down design elaboration.
- Detects include dependencies: If a (S)Verilog file is included in another source, and the file changes between two consecutive calls to the **-autoread** option (in the same DCSHELL session), this file and all files that include them will be scheduled for analyze to update the design.
- Infers the target library for VHDL files. However, if the **-library** option is specified, this step is skipped and VHDL files provided in *file_list* are analyzed into the specified design library.

After the dependency inference, all the required HDL files from the *file_list* (regarding its dependencies and the **-top** design defined) are analyzed, and then the design is elaborated from the top-level module. If no elaboration is desired, you can use the **analyze -autoread** command. The resulting GTECH representation is retained in memory.

It is possible to save the design to disk by using the **write** command.

When the **-autoread** option is used again after a file is changed, only the updated source files are analyzed and the design is elaborated. The updated designs are replaced in the unmapped design.

The **-autoread** option executes the dependency analysis only based on the current *file_list* information. Therefore, it is recommended to pass all RTL files that might have relevant dependency relationship in a single **read_file -autoread** command.

EXAMPLES

The following example uses simple and complex file names to read in a file of equations. This is a complex path specification, so **search_path** is not used.

```
prompt> read_file -format equation /usr/test/test.fnc
Loading file '/usr/test/test.fnc'
{test}
```

The following example uses a simple file name to specify the same file as shown in the previous example. Thus, its directory must appear in the **search_path**.

```
prompt> search_path = {/usr/test}
{/usr/test}
prompt> read_file -format equation test.fnc
Loading file '/usr/test/test.fnc'
{test}
```

The following example reads a file that contains multiple designs, so each design is assigned its own internal design file name. The asterisk (*) indicates the current design. For more information, see the **list_designs** man page.

```
prompt> read_file -format verilog foo.v
Loading file '/usr/test/foo.v'
{A B C}
```

```
prompt> list_designs
```

	Design	File	Path
	-----	----	----
*	A	A.ddc	/usr/test/dc
	B	B.ddc	/usr/test/dc
	C	C.ddc	/usr/test/dc

The following example shows that when you use the **-single_file** option, all designs are assigned the same file name:

```
prompt> read_file -format verilog -single_file mylib.ddc foo.v
Loading file '/usr/test/foo.v'
{A B C}
```

```
prompt> list_designs
```

	Design	File	Path
	-----	----	----
*	A	mylib.ddc	/usr/bill/dc
	B	mylib.ddc	/usr/bill/dc
	C	mylib.ddc	/usr/bill/dc

The following example uses simple and complex file names to read the library .db file into the shell. This is a complex path specification, so the **search_path** variable is not used.

```
prompt> read_file -format db /usr/test/test.db
Loading db file '/usr/test/test.db'
{test}
```

The following example shows that in order to specify the same file as shown in the example above as a simple file name, its directory must appear in the **search_path** variable.

```
prompt> search_path = {/usr/test}
{/usr/test}

prompt> read_file -format db test.db
Loading db file '/usr/test/test.db'
{test}
```

The following example assumes that the current directory is the source directory. It specifies the source file list at the command line and calls the command with the name of the top-level entity.

```
prompt> read_file {.} -autoread -recursive -top E1
```

The next example specifies extensions for Verilog files that are different than the default ({.v}), sets the source list and the exclude list, and calls the command with the name of the top-level module name, forcing it to only collect files with Verilog extensions.

```
prompt> set hdlin_autoread_verilog_extensions {.ve .VE}
prompt> set my_sources {mod1/src mod2/src}
prompt> set my_excludes {mod1/src/incl mod2/src/incl}
prompt> read_file $my_sources -exclude $my_excludes \
-autoread -format verilog -top TOP
```

Note that excluding include directories explicitly is only necessary if include files have the same extensions as source files and not all include files are included in the source.

SEE ALSO

analyze(2)
all_scenarios(2)
all_active_scenarios(2)
change_names(2)
create_scenario(2)
elaborate(2)
link(2)
list_designs(2)
list_files(2)
list_libs(2)
set_active_scenarios(2)
which(2)
write(2)
search_path(3)
view_read_file_suffix(3)
UID-1033(n)
UID-1034(n)
UID-1328(n)
hdlin_autoread_exclude_extensions(3)
hdlin_autoread_verilog_extensions(3)
hdlin_autoread_sverilog_extensions(3)
hdlin_autoread_vhdl_extensions(3)

read_floorplan

Reads a script that describes a floorplan into the current design.

SYNTAX

```
status read_floorplan
    file_name
    [-echo]
    [-verbose]
```

Data Types

file_name string

ARGUMENTS

file_name

Specifies the name of the file that contains the floorplan script to be read.

-echo

Specifies that each command is written out as it is executed.

-verbose

Displays the result of each command executed. Note that error messages are displayed regardless. Also note that this option is a non-standard extension to Tcl.

DESCRIPTION

The **read_floorplan** command reads in a script file that contains commands describing floorplan information.

Similar to the **source** command, the **read_floorplan** command is guided by variables that control script execution, such as the **sh_continue_on_error** and **sh_script_stop_severity** commands. The **read_floorplan** command uses the **sh_source_uses_search_path** variable to determine if the **search_path** command is used to find the script file.

By default, **read_floorplan** displays only error messages and summary output. It is possible to get additional information from the **read_floorplan** command using the **-echo** and **-verbose** options. The **-echo** option writes out each command as it appears in the script. The **-verbose** option echoes the result of each command after execution.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reads the floorplan script named *top.fp*:

```
prompt> read_floorplan top.fp  
Reading floorplan ...  
1
```

SEE ALSO

- source(2)
- write_floorplan(2)
- search_path(3)
- sh_continue_on_error(3)
- sh_script_stop_severity(3)
- sh_source_uses_search_path(3)

read_lib

Reads a technology library into the shell.

Starting with the K-2015.06 release, the Library Compiler tool is a separate installation that is no longer part of the synthesis tools installation. Therefore, to run the Library Compiler **read_lib** command in the Design Compiler tool, the two tools must be linked during the tool installation process. For information about the installation process and how to link the Library Compiler tool to the Design Compiler tool, see the Synthesis Tools Installation Notes and the Library Compiler Installation Notes for the K-2015.06 release.

SYNTAX

```
status read_lib
    file_name
    [-test_model CTL_file_list]
    [-html]
    [-symbol slib_filename]
    [-names_file file_list]
    [-imported_jcr jcr_file]
    [-format format]
    [-lib_messages lib_msgs]
    [-plibrary plibrary_file_name]
    [-no_warnings]
    [-pplibrary -pplibrary_filename]
```

Data Types

<i>file_name</i>	string
<i>CTL_file_list</i>	list

ARGUMENTS

file_name

Specifies the name of the library file to be read. A technology must be in Synopsys Library Compiler format.

-test_model *CTL_file_list*

Specifies a set of one or more CTL files as test model of cells in the library. The format of the *CTL_file_list* is *<cell_name>:file_name* where *cell_name* is optional and is to specify the particular cell a CTL file models. When the *cell_name* is missing, the environment name in the CTL file is used as the cell name it is modeling. When specifying more than one CTL file, enclose them in braces ({}).

-html

Specifies that library screener results are reported in HTML.

DESCRIPTION

The **read_lib** command reads a library file into the shell (or GUI). The library file is automatically compiled by the Synopsys Library Compiler. The library specified in *file_name* can have either a simple or complex file name. A simple file name has no directory specification, which means (in UNIX) that it does not contain a / (slash). Simple file names such as test.lib or library.db must be found in a directory listed in the **search_path** variable. A complex file name has a directory specification in it, which means that it does contain a slash. Complex file names such as ./test.lib or ~synopsys/dc/test.lib are not searched for in **search_path**.

You can read any number of libraries into the shell. Two libraries can have the same name in the system as long as they are read from different sources. The **list_libs** command shows all libraries present in the system.

The **read_lib** command will check the input model file library syntax against model syntax specified by **-model_type** option. If any content of the input model file library does not belong to the specified model, syntax error will be reported.

For details on library file syntax, see the Library Compiler documentation. Function statements and state information are ignored when reading technology library files if you do not have a Library Compiler license.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, the library file *tech_lib.lib*, is read. This file contains technology information in Synopsys technology library entry format,

```
prompt> read_lib ~synopsys/libraries/tech_lib.lib
```

The following example reads a library file as well as a CTL file when the **-test_model** option is used:

```
prompt> read_lib bar.lib -test_model {usr1.ctl usr2:usr2.ctl}
```

The following examples read in a lvf, ccs_power or power_aware model file respectively when the **-model_type** option is used:

```
prompt> read_lib aocv.lib -model_type lvf
prompt> read_lib aocv.lib -model_type ccs_power
prompt> read_lib aocv.lib -model_type power_aware
```

The following example reads a library file and report screener messages in HTML:

```
prompt> read_lib bar.lib -html
```

SEE ALSO

write_lib(2)
search_path(3)

read_ocvm

Reads a parametric on-chip variation (POCV) derating factor table.

SYNTAX

```
status read_ocvm  
  [-min]  
  [-max]  
  [-distance_row row_number]  
  ocvm_file
```

Data Types

```
row_number  integer  
ocvm_file   string
```

ARGUMENTS

-min

Specifies that the POCV derating table applies only to the minimum operating condition.

-max

Specifies that the POCV derating table applies only to the maximum operating condition.

By default, the POCV derating table applies to both the minimum and maximum operating conditions.

-distance_row *row_number*

This option is not used in the Design Compiler tool.

ocvm_file

Specifies the name of the POCV derating table file.

DESCRIPTION

The **read_ocvm** command reads POCV derating tables from a disk file. The tables are annotated onto one or more design objects: hierarchical cells, library cells, or designs. The POCV data is used to reduce pessimism and improve the accuracy of timing results by applying different derating factors based on the depth of each path (the number of successive gates in the path).

To use the POCV files read by the **read_ocvm** command and perform POCV analysis, set the **timing_pocvm_enable_analysis** variable to **true**.

To apply the table to only to the maximum operating condition, use the **-max** option when you run the **read_ocvm** command.

Multicorner-Multimode Support

This command uses information from the current scenario only. By default, if the design objects are library cells, the tables are applied to all scenarios. To apply the tables only to the current scenario, set the **timing_library_derate_is_scenario_specific** variable to **true** before you apply the tables to the library cells.

EXAMPLES

In the following example, the first command shows a POCV table file named test.pocvm, which contains a single table to be annotated on the design. The second command reads in and applies this table.

```
prompt> sh cat test.pocvm
```

```
version:    4.0
```

```
object_type: design
```

```
rf_type:    rise
```

```
delay_type: cell
```

```
derate_type: late
```

```
object_spec:
```

```
coefficient: 0.05
```

```
prompt> read_ocvm test.pocvm
```

SEE ALSO

[remove_ocvm\(2\)](#)

[report_ocvm\(2\)](#)

[report_timing\(2\)](#)

[timing_pocvm_enable_analysis\(3\)](#)

[timing_library_derate_is_scenario_specific\(3\)](#)

read_parasitics

Reads net parasitics information from an SPEF, DSPF, or RSPF file, and uses it to annotate the current design.

SYNTAX

```
status read_parasitics
  [-syntax_only]
  [-elmore | -arnoldi]
  [-increment]
  [-pin_cap_included]
  [-net_cap_only]
  [-complete_with zero | wlm]
  [-path path_name]
  [-strip_path path_name]
  [-quiet | -verbose]
  [-dont_write_to_db]
  [file_list]
```

Data Types

```
path_name  string
file_list  list
```

ARGUMENTS

-syntax_only

Checks the parasitic data files for errors and reports any errors without actually applying the data to the design.

-elmore

Causes the tool to estimate delays based on the parasitics information using the Elmore delay model and to back-annotate the estimated delays, in addition to creating the RC tree. If neither **-elmore** or **-arnoldi** is specified, the RC tree is created without estimating or back annotating delays.

-arnoldi

Causes the tool to estimate delays based on the parasitics information using the Arnoldi delay calculator and to back-annotate the estimated delays, in addition to creating the RC tree. If neither **-elmore** nor **-arnoldi** is specified, the RC tree is created without estimating or back-annotating delays.

-increment

Retains any previously annotated parasitics on the nets listed in the parasitics file. This option is used for annotating hierarchical parasitics files. By default, the RC annotation specified in the parasitics file overwrites the previous parasitics annotations of the

nets listed in the parasitics file.

-pin_cap_included

Specifies that the RC network includes the pin capacitance. By default, the RC network does not include them.

-net_cap_only

Back-annotates only the net capacitance values, not the pin-to-pin delay values. By default, both the capacitance values and the pin-to-pin delays are back-annotated.

-complete_with zero | wlm

Causes the tool to automatically complete the RC tree after parsing. This option should be used if a parasitics file contains incomplete parasitics information; for example, if only a segment of a wire is annotated. During the RC completion phase, missing wire segments are automatically added.

If **zero** is specified, all newly added segments and nodes are considered to have 0 resistance and 0 capacitance. If **wlm** is specified, the capacitance and resistance values for a new segment are determined by the current wireload model.

-path path_name

Specifies the path from the current design to the specified subdesign for which the parasitics file has been created.

-strip_path path_name

Strips off the specified prefix path name from all object names in the parasitics file. This is usually a result of generating a parasitics file for a subdesign, and then using this subdesign as the current design.

-quiet

Skips the implicit **report_annotated_delay** report when the parasitics file has been read. By default, after reading the parasitics file, the **report_annotated_delay** command is executed automatically, which reports the back-annotated delays of all nets.

-verbose

Generates an annotation report after parsing.

-dont_write_to_db

Suppresses writing back to the Synopsys database (.db file). Use this option to save runtime when you do not need the parasitic data in the .db database.

file_list

Specifies a list of files from which to read parasitics information. The supported formats are Standard Parasitic Exchange Format (SPEF), Detailed Standard Parasitics Format (DSPF), and Reduced Standard Parasitics Format (RSPF).

DESCRIPTION

The **read_parasitics** command reads parasitics information from an disk file and back-annotates the data on the nets of the current design.

You can specify parasitics in either a reduced form or a detailed form:

- Reduced parasitics consist of an RC pi model specified for each driver pin of a net. An RC pi model contains two capacitors and one resistor. The first capacitor connects to the net driver pin. The resistor connects to the net driver pin and to a subnode to which the second capacitor connects. Both capacitors connect to ground. The parasitics file using the reduced form also

specifies the pin-to-pin net delays.

- Detailed parasitics consist of a network of resistors and capacitors for each net specified in the parasitics file. The capacitors are with respect to ground. Coupling capacitors are not supported. Resistors between the nets and ground are ignored. Delays are computed via the Elmore delay model (using the **-elmore** option) or the Arnoldi delay model (using the **-arnoldi** option).

Net and instance pin names in the design must match instance names in the parasitics file. For example, if you create the parasitics file from a design using VHDL naming conventions, the design names must also use the same conventions.

To remove loads and delays annotated by using the **read_parasitics** command, use the **reset_design** or **remove_annotated_delay** command.

Multicorner-Multimode Support

This command uses information from the current scenario only. Define all scenarios with TLUPlus files and operating conditions settings before using this command. The parasitics files reading order must be the same as the scenario creation order.

EXAMPLES

The following command reads the parasitics file named `adder.spf` from disk and uses it to annotate the computed Elmore delays to pins and loads to nets. The parasitics file contains a description of the RC network for nets of a design.

```
prompt> read_parasitics adder.spf -elmore
```

The following example removes annotated delays and loads from the current design:

```
prompt> remove_annotated_delay
```

SEE ALSO

```
remove_annotated_delay(2)  
report_annotated_delay(2)  
reset_design(2)  
write_parasitics(2)
```

read_partition

Loads a Verilog netlist into a partition editor

SYNTAX

```
status read_partition  
  [-format db | ddc]  
  design_name  
  -source pass  
  -type pre | post
```

ARGUMENTS

DESCRIPTION

This command allows you to load a Verilog netlist into a partition editor. After loading the netlist, you can use the **set_partition_data** and **create_partition** commands to determine a partitioning for the design and to create a Milkyway database for the design.

EXAMPLES

The following example creates a partition editor:

```
prompt> read_partition -input_files {myDesign.v}  
Partition Name: pfe0  
Loading verilog file myDesign.v  
MYDESIGN: 73958 instances of 134 modules.  
1
```

SEE ALSO

read_pin_map

Reads in a port-to-pin mapping file, which defines the design port-to-package pin mapping for a boundary-scan design.

SYNTAX

```
status read_pin_map  
    path_name
```

Data Types

```
path_name    string
```

ARGUMENTS

path_name

Specifies the full name of the file containing the port-to-pin mapping for the design.

DESCRIPTION

The **read_pin_map** command allows you to read in a port-to-pin mapping file for a boundary-scan design. All the signal ports in the design must be listed in the port-to-pin mapping file. If a port listed in the port-to-pin mapping file is not a port of the design, it is assumed to be a linkage port. The **write_bsd** command requires the design port-to-package pin mapping information in order to generate a boundary-scan description language (BSDL) file for a boundary-scan design.

To remove all port-to-pin mapping for a design, use the **remove_pin_map** command. To list all the port-to-pin mappings read in for the design, use the **report_packages** command.

EXAMPLES

The following is an example of a port-to-pin mapping file for a design:

```
PACKAGE = test_pkg;  
  
PORT = OUT5, PIN = (P41, P40, P39, P38, P37, P36);  
PORT = OUT2, PIN = (P26, P27, P28, P29, P30, P31);  
PORT = IN1, PIN = P1;
```

```
PORT = IN2, PIN = P2;  
PORT = IN3, PIN = P3;  
PORT = EN, PIN = P4;  
PORT = RESETN, PIN = P5;  
PORT = IN4, PIN = P6;  
PORT = CLK, PIN = P7;  
PORT = CLOCKDR, PIN = P8;  
PORT = configENABLE, PIN = P9;  
PORT = jtag_tdi, PIN = P10;  
PORT = jtag_tdia, PIN = P11;  
PORT = jtag_tms, PIN = P12;  
PORT = jtag_tck, PIN = P13;  
PORT = jtag_trst, PIN = P14;  
PORT = configCAPTUREx, PIN = P15;  
PORT = configENABLEx, PIN = P16;  
PORT = OUT3, PIN = P17;  
PORT = OUT6, PIN = P18;  
PORT = OUT6enable, PIN = P19;  
PORT = configCAPTURE, PIN = P20;  
PORT = jtag_tdo, PIN = P21;  
PORT = OUT1(3), PIN = P22;  
PORT = OUT1(2), PIN = P23;  
PORT = OUT1(1), PIN = P24;  
PORT = OUT1(0), PIN = P25;  
PORT = OUT4(3:0), PIN = (P32, P33, P34, P35);  
PORT = CONFIGURATION(3:0), PIN = (P42, P43, P44, P45);  
PORT = VDD, PIN = (P46, P48, P50);  
PORT = GND, PIN = (P47, P49, P51);
```

The following example reads in a design port-to-package pin mapping file for a boundary-scan design:

```
prompt> read_pin_map bscan.map
```

SEE ALSO

```
get_attribute(2)  
remove_pin_map(2)
```

read_saif

Reads a SAIF file and annotates switching activity information on nets, pins, ports, and cells in the current design.

SYNTAX

```
status read_saif
-input file_name
[-instance_name name]
[-target_instance instance]
[-ignore ignore_name]
[-ignore_absolute ig_absolute_name]
[-exclude exclude_file_name]
[-exclude_absolute ex_absolute_file_name]
[-names_file name_changes_log_file]
[-scale scale_value]
[-unit_base unit_value]
[-khrate khrate_value]
[-map_names]
[-auto_map_names]
[-verbose]
```

Data Types

<i>file_name</i>	string
<i>name</i>	string
<i>instance</i>	string
<i>ignore_name</i>	string
<i>ig_absolute_name</i>	string
<i>exclude_file_name</i>	string
<i>ex_absolute_file_name</i>	string
<i>name_changes_log_file</i>	string
<i>scale_value</i>	integer
<i>unit_value</i>	string
<i>khrate_value</i>	float

ARGUMENTS

-input *file_name*

Specifies the name of the SAIF file to be read. When the *file_name* ends with the *.gz* extension, the input file will be uncompressed automatically.

-instance_name *name*

Specifies the name of the instance of the current design as it appears in the SAIF file. The **read_saif** command assumes that the current design is instantiated as an instance, in the testbench used to generate the SAIF file. The current design appears as an instance in the SAIF file. The **read_saif** command annotates all subinstances in the hierarchy of the specified instance, and annotates the instance itself. Each instance name that you specify must be complete, but without any trailing hierarchy separator (/).

-target_instance *instance*

Specifies the target instance on which the switching activity, in the SAIF file, is to be annotated. If this option not specified, the current instance is assumed to be the target instance.

-ignore *ignore_name*

Specifies the name of an instance in the SAIF file for which switching activity is to be ignored. The **read_saif** command ignores switching activity within the hierarchy of that instance in the SAIF file. The *-ignore* argument can be a hierarchical name separated by a slash (/). The **-instance_name** is applied first. The **read_saif** command then strips off the prefix of a net name that matches the *name*, before deciding whether or not this net name is under the hierarchy specified by the **-ignore** option.

-ignore_absolute *ig_absolute_name*

Specifies the name of an instance in the SAIF file for which switching is ignored. But this option is not affected by the **-instance_name** argument. The **read_saif** command determines whether a net name is under the **-ignore** hierarchy before applying the **-instance_name** argument.

-exclude *exclude_file_name*

Specifies the name of a file that contains a list of names to be ignored. The **-exclude** option is useful when you want to ignore switching activity for several instances. The file must contain each *ignore_name* on a separate line, without the **-exclude** option. The ignored names are recognized after the **-instance_name** argument.

-exclude_absolute *ex_absolute_file_name*

Specifies the name of a file that contains a list of absolute names to be ignored. The absolute names are not affected by the **-instance_name** option.

-names_file *name_changes_log_file*

Specifies a previously created log file, which is usually the output of a **change_names -log_changes** command. The log file is used as the input to the **read_saif** command. While the design file can contain multiple name changes, the SAIF file contains only the original object names. So the object search by name might fail to find objects with names that were changed. The *name_changes_log_file* contains a list of **change_names** commands for cells, ports, and nets in the current design. The file provides mapping information so that **read_saif** can identify the correct objects, even if the names have changed. Note that *name_changes_log_file* is a text file, and it can be edited manually to include other name changes that are not captured by the **change_names** command.

-scale *scale_value*

Specifies the value of the scaling factor for the base synthesis time unit. Allowed values for *scale_value* are **1**, **10**, and **100**. **1** is the default value. The synthesis time unit is usually obtained automatically from the target library. In some cases when the time unit in the target library is unknown, the **read_saif** command issues a warning message that the intended synthesis time unit is assumed to be 1 ns (nanosecond). For example, this can occur when you run the **read_saif** command before running the **compile_ultra** command. If you see this warning message and the time unit in the intended target library is not 1 ns, then use the **-scale** and **-unit_base** options to specify the time unit of the intended target library. For example, if the intended time unit is 100 ps, specify **-scale 100 -unit_base ps**.

-unit_base *unit_value*

Specifies the base synthesis time unit. Allowed values for *unit_value* are **ns**, **us**, or **ps**. **ns** is the default. See the details of the **-scale** option for information on how to use it with the **-unit_base** option.

-khrate *khrate_value*

Specifies a default derating factor value to use for inertial glitches in the SAIF file. If **-khrate** is not specified, the tool uses the default derating factor of 0.5.

-map_names

Specifies that the SAIF name mapping mechanism is used to match the objects in the SAIF file with the design objects.

-auto_map_names

Specifies that a name mapping is to be created automatically using the SAIF file, and the created name mapping is to be used to read the SAIF file. The **-auto_map_names** flag has the effect of creating the name mapping information using the **saif_map -create_map** command and reading in the SAIF file information using the **read_saif -map_names** command.

-verbose

Shows detailed warnings when design objects in the SAIF file cannot be annotated on the design.

OBSOLETE ARGUMENTS

-rtl_direct

The flows involving forward RTL SAIF files, and flows involving the **read_saif -rtl_direct** is obsolete.

DESCRIPTION

The **read_saif** command reads a SAIF file, and annotates the switching activity attributes, **toggle_rate** and **static_probability** for nets, ports, and pins of the current design. The **report_power** command uses this information to calculate dynamic power values. If a particular object in the SAIF file is not found in the current design, the tool ignores the object and issues a warning message. The **read_saif** command returns 1 if at least one of the objects in the file is successfully annotated. Otherwise, it returns 0.

To specify the instance (and corresponding subinstances) you want to annotate, use the **-instance_name** option. To set the synthesis time unit, use **-unit_base** and **-scale**, unless you are certain that the synthesis time unit is 1 ns. To specify a default derating factor other than 0.5, use the **-khrate** option.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example annotates the switching activity on the current design, that has been instantiated as Test/U1 in the simulation testbench:

```
prompt> read_saif -input file -instance_name Test/U1
```

The following example reads directly from a compressed file and annotates the switching activity on the current design, that has

been instantiated as Test/U1 in the simulation testbench:

```
prompt> read_saif -input file.gz -instance_name Test/U1
```

The following commands annotate the current design, that has been instantiated as Test/U1 in the simulation testbench, assuming that the library time unit is 10 ns:

```
prompt> read_saif -input file -instance Test/U1 -scale 10 \
-unit ns
```

```
prompt> read_saif -input file -instance Test/U1 -scale 10
```

The following commands demonstrate annotation of multiple designs. The *demo1* design is instantiated as Test1/U1, and the *demo2* design is instantiated as Test1/U2.

```
prompt> current_design mydesign
```

```
prompt> read_saif -input file_1 -instance Test1/U1 -scale 10
```

```
prompt> current_design demo2
```

```
prompt> read_saif -input file_2 -instance Test1/U2 -scale 10
```

The following examples illustrate the use of the **-names_file** option. In the first three commands, the name_changes.log file is created. In the fourth command, the name_changes.log file is used as input to the **read_saif** command. The name changes recorded in the name_changes.log file are used in the name-matching during the execution of the **read_saif** command.

```
prompt> read_ddc design.ddc
```

```
prompt> define_name_rules my_rules -restricted "A-Z"
```

```
prompt> change_names -rule my_rules -log_changes name_changes.log
```

```
prompt> read_saif -input file -instance Test/U1 \
-names name_changes.log
```

SEE ALSO

- report_saif(2)
- report_power(2)
- saif_map(2)
- set_switching_activity(2)

read_scan_def

Reads scan chain information from a SCANDEF file.

SYNTAX

```
int read_scan_def  
    file_name
```

Data Types

file_name string

ARGUMENTS

file_name

Specifies the name of the input SCANDEF file.

DESCRIPTION

The **read_scan_def** command reads in scan chain information from a SCANDEF file. Once the SCANDEF file is read, the tool automatically runs **check_scan_def** to validate that all scan chains are properly read.

The **read_scan_def** command allows you to optimize a design read in as a gate-level netlist while preserving its scan chain structure. After optimization, you can write out the up-to-date SCANDEF file by using the **write_scan_def** command.

EXAMPLES

The following example reads a gate-level design from the source file top.v and then reads the corresponding SCANDEF file top.scandef for the current design top. After optimization, the up-to-date SCANDEF file is written out with the **write_scan_def** command.

```
prompt> read_verilog -netlist top.v  
prompt> current_design top  
prompt> read_scan_def top.scandef  
prompt> compile_ultra -incr  
prompt> write_scan_def -output top.incr.scandef
```

```
prompt> write -f verilog -h -o top.incr.v
```

SEE ALSO

[write_scan_def\(2\)](#)
[check_scan_def\(2\)](#)

read_sdc

Reads in a script in Synopsys Design Constraints (SDC) format.

SYNTAX

```
status read_sdc  
  file_name  
  [-echo]  
  [-syntax_only]  
  [-version sdc_version]
```

Data Types

<i>file_name</i>	string
<i>sdc_version</i>	string

ARGUMENTS

file_name

Specifies the name of the file that contains the SDC script to be read.

-echo

Indicates that each constraint is to be echoed as it is executed.

-syntax_only

Indicates that SDC script is processed only to check the syntax and semantics of the script.

-version *sdc_version*

Specifies the version of SDC to which the file conforms. Allowed values are 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1 and latest (the default).

DESCRIPTION

This command reads in a script file in Synopsys Design Constraints (SDC) format, which contains commands for use by PrimeTime or by Design Compiler in its Tcl mode. SDC is also licensed by external vendors through the Tap-in program. SDC formatted script files are Tcl scripts that use a subset of the commands supported by PrimeTime and Design Compiler.

By default, **read_sdc** executes the constraints as they are read. If errors occur before the script completes, it is possible that some

constraints are applied, and some are not. You can use the **-syntax_only** option to check the script without applying any constraints. This also verifies conformance to the specified SDC version. If there are any errors during the checking phase, then the result of **read_sdc** is 0.

Like the **source** command, the **read_sdc** command is sensitive to variables that control script execution (for example, **sh_continue_on_error** and **sh_script_stop_severity**). The **read_sdc** command uses **sh_source_uses_search_path** to determine if **search_path** is used to find the script.

The **read_sdc** command supports several file formats. The file can be a simple ASCII script file, an ASCII script file compressed with gzip, or a Tcl bytecode script, created by TclPro Compiler.

Note that SDC does not allow command abbreviation. Also note that the **exit** and **quit** commands do not cause the application to exit, but they do cause the reading of the SDC file to stop.

The latest SDC Version supports the following commands. Those added for versions 1.3 and later are noted. Some constraints that PrimeTime ignores are not listed.

General Purpose Commands

list
expr
set

Object Access Functions

all_clocks
all_inputs
all_outputs
current_design
current_instance
get_cells
get_clocks
get_libs
get_lib_cells
get_lib_pins
get_nets
get_pins
get_ports
set_hierarchy_separator

Basic Timing Assertions

create_clock
create_generated_clock (1.3)
set_clock_gating_check
set_clock_latency
set_clock_transition
set_clock_uncertainty
set_data_check (1.4)
set_false_path
set_input_delay
set_max_delay
set_min_delay
set_multicycle_path
set_output_delay
set_propagated_clock
set_sense (2.1)

Secondary Assertions

set_disable_timing

set_max_time_borrow
set_timing_derate (1.5)

Environment Assertions

set_case_analysis
set_drive
set_driving_cell
set_fanout_load
set_input_transition
set_load
set_logic_zero
set_logic_one
set_logic_dc
set_max_area
set_max_capacitance
set_max_fanout
set_max_transition
set_min_capacitance
set_min_fanout
set_operating_conditions
set_port_fanout_number
set_resistance
set_wire_load_min_block_size
set_wire_load_mode
set_wire_load_model
set_wire_load_selection_group

For a complete guide to using SDC with Synopsys applications, see the *Using the Synopsys Design Constraints Format Application Note* which is available through SolvNET at <http://solvnet.synopsys.com>.

The usage of some of these commands is restricted when reading SDC. In some cases, some command options are not allowed. In some applications, some of the commands are not supported. For example, PrimeTime does not support the **set_logic*** commands. These commands are ignored when loaded in with **read_sdc**, with a message (for an example, see the EXAMPLES section). If a command that is not supported by SDC is found by **read_sdc**, it appears as an unknown command, with a message. The same condition exists for command options. If an option is not supported by SDC, a message is issued indicating that condition. However, if the option is unknown, a different message is issued.

Note that although **create_generated_clock** is supported (beginning with version 1.3), there is no provision for the **get_generated_clocks** shortcut, which is available in PrimeTime and Design Compiler. Generated clocks are simply clocks with additional attributes and in SDC, they are retrieved through the **get_clocks** command.

The following features are added for SDC Version 1.5 or later:

- **set_clock_latency -clock** option
- **set_timing_derate** command with all options
- **set_max_transition -clock_path -data_path -rise -fall** options
- **set_clock_uncertainty -rise/fall_from/to** options
- **set_operating_conditions -object_list** option synonyms for all **get_object** commands
- **get_objects -nocase** support independently with **-regexp**
- **get_objects -regexp** support, **get_objects -of_objects** support for **get_cells/get_nets/get_pins**

The following features are added for SDC Version 2.1 or later:

- The **set_sense** command with the *-type*, *-non_unate*, *-positive*, *-negative*, *-stop_propagation*, *-clock_leaf*, *-pulse*, *-clocks* options.
- The *-dynamic* option for the command **set_clock_latency**.
- The *-ignore_clock_latency* option for the command **set_max_delay**.
- The *-ignore_clock_latency* option for the command **set_min_delay**.

The following features are removed for SDC Version 2.1 or later, and are not recognized:

- The **set_clock_sense** command.
- The *-multiply_by* option for the command **set_driving_cell**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example reads the SDC script named *top.sdc*:

```
prompt> read_sdc top.sdc -version 1.2
```

```
Reading SDC version 1.2...
```

```
1
```

The following example reads the SDC script *top2.sdc*. The script contains commands not supported by SDC, and CMD-005 messages are generated for those commands.

```
prompt> read_sdc -syntax_only top2.pt -version 1.2
```

```
Checking syntax/semantics using SDC version 1.2...
```

```
Error: Unknown command 'link_design' (CMD-005)
```

```
** Completed syntax/semantic check with 1 error(s) **
```

```
0
```

The following example shows the result when an unsupported command is in an SDC file. One SDC message is generated per instance of an unsupported command. At the end of the read, a summary of all unsupported commands in the file is generated. This SDC file was read into PrimeTime.

```
prompt> read_sdc t2.sdc -version 1.2
```

```
Reading SDC version 1.2...
```

```
Warning: Constraint 'set_logic_zero' is not supported by pt_shell. (SDC-3)
```

```
Warning: Constraint 'set_logic_zero' is not supported by pt_shell. (SDC-3)
```

```
Warning: Constraint 'set_logic_one' is not supported by pt_shell. (SDC-3)
```

```
Summary of unsupported constraints:
```

```
Information: Ignored 1 unsupported 'set_logic_one' constraint. (SDC-4)
```

```
Information: Ignored 2 unsupported 'set_logic_zero' constraints. (SDC-4)
```

```
1
```

The following example is a script that is not SDC-compliant, because it contains unsupported commands or command options:

```
current_design TOP
```

```
set_resistance -value 12.2 [get_nets i1 t2]
```

The following example shows the output generated by **read_sdc** when reading the previous script. In SDC, **current_design** can be used only to get the current design; no arguments are allowed. Also, there is no **-value** option for **set_resistance**, so an appropriate message is generated.

```
prompt> read_sdc t3.tcl -echo -version 1.2
Reading SDC version 1.2...
current_design TOP
Error: extra positional option 'TOP' (CMD-012)
set_resistance -value 12.2 [get_nets i1 t2]
Error: unknown option '-value' (CMD-010)
0
```

The following example shows the message generated when an option is supported by the command but not by SDC:

```
Information: The '-value' option for set_resistance is unsupported. (CMD-038)
```

SEE ALSO

```
source(2)
write_sdc(2)
search_path(3)
sh_continue_on_error(3)
sh_script_stop_severity(3)
sh_source_uses_search_path(3)
```

read_sdf

Reads leaf-cell and net-timing information from a file in Standard Delay Format (SDF) and uses that information to annotate the current design.

SYNTAX

```
string read_sdf  
  [-load_delay net | cell]  
  [-path path_name]  
  [-min_type sdf_min | sdf_typ | sdf_max]  
  [-max_type sdf_min | sdf_typ | sdf_max]  
  [-worst]  
  [-min_file min_sdf_file_name]  
  [-max_file max_sdf_file_name]  
  sdf_file_name
```

Data Types

```
path_name      string  
min_sdf_file_name string  
max_sdf_file_name string  
sdf_file_name  string
```

ARGUMENTS

-load_delay net | cell

Indicates whether load delays are included in net delays or in cell delays in the timing file being read. The load delay is the portion of the cell delay arising from the capacitive load of the net driven by the cell. The default is **cell**.

-path *path_name*

Specifies the path from the current design to the subdesign for which the timing file has been created.

-min_type sdf_min | sdf_typ | sdf_max

Specifies which of the SDF triplet delay values are to be read from the SDF file for minimum delay. Delays in SDF are represented in the form of triplets (sdf_min:sdf_typ:sdf_max). If this is not specified, the command uses **sdf_min**.

-max_type sdf_min | sdf_typ | sdf_max

Specifies which of the SDF triplet delay values are to be read from the SDF file for maximum delay. Delays in SDF are represented in the form of triplets (sdf_min:sdf_typ:sdf_max). If this is not specified, the command uses **sdf_max**.

-worst

Indicates that **read_sdf** is to annotate the current design only with delays worse than the current annotated delays; this applies to annotated net and cell delays and annotated timing checks. For the hold timing check, the minimum numerical value is annotated after comparing the current annotated value and the new value. For all other timing checks and delays, the maximum numerical value is annotated after comparing the current annotated value and the new value. Use this option when the timing file contains conditional timing.

To annotate the worst timing delay and timing checks of all timing conditions for each pin-to-pin annotation, use the **remove_annotated_delay** command with the **-all** option and the **remove_annotated_check** command before using the **read_sdf** command with the **-worst** option. Note that this behavior does not depend on the type of timing arc being read (that is, whether the arc is a setup or a hold arc), even though it might appear that minimum values are read for hold timing arcs. The correct behavior is one where all the values read from SDF files are either minimum, typical, or maximum, regardless of the type (setup or hold).

-min_file min_sdf_file_name

Specifies the file from which minimum delay timing information is to be read. The timing file must be in SDF format version v1.0, v2.0, or v2.1. Use this option only if the minimum and maximum delays are in two separate SDF files. The default is for minimum and maximum delays to be in a single SDF file.

-max_file max_sdf_file_name

Specifies the file from which maximum delay timing information is to be read. The timing file must be in SDF format version v1.0, v2.0, or v2.1. Use this option only if the minimum and maximum delays are in two separate SDF files. The default is for minimum and maximum delays to be in a single SDF file.

sdf_file_name

Specifies the name of the file from which the timing information is to be read. The timing file must be in SDF format version v1.0, or v2.1.

DESCRIPTION

This command reads from a disk file SDF leaf-cell and net-timing information and uses it to annotate the current design. The timing file must be in SDF format v1.0 or v2.1. Instance-specific pin-to-pin cell and net delays are read from the SDF file and annotated on the current design. When you specify the **-path** option, the command annotates the current design with information from a timing file created for an instance of a subdesign of the current design. When you specify a subdesign, you cannot annotate the current design with the net delays to the ports of the subdesign.

The load delay, also known as extra source gate delay, is that portion of the cell delay caused by the capacitive load of the driven net. Some delay calculators consider the load delay part of the net delay; others consider the load delay part of the cell delay. By default, the **read_sdf** command treats the load delay as part of the cell delay in the timing file being read. Use the **-load_delay net** option to indicate that your timing file includes the load delay in the net delay instead of in the cell delay.

Setup and hold, recovery, and removal timing checks, when present in the timing file, are used to annotate the current design. The SDF constructs for setup and hold are "SETUP" and "HOLD". The SDF construct for recovery and removal are "RECOVERY", "REMOVAL" and "RECREM." \" replace sentences above with the following when SDFv3.0 is supported: Instance names in the design must match instance names in the timing file. For example, if the timing file is created from a design using VHDL naming conventions, the design must use VHDL naming conventions. To modify design names, use the **change_names** command.

To remove delays read and annotated by using the **read_sdf** command, use the **reset_design** or **remove_annotated_delay** command. To remove timing checks annotated with **read_sdf**, use the **remove_annotated_check** command.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example reads the *adder.sdf* timing file from disk and uses it to annotate the timing on the current design. The timing file contains load delays included in the cell delays.

```
prompt> read_sdf -load_delay cell adder.sdf
```

The following example reads the timing information of instance *u1* of design *MULT16* from the *mult16_u1.sdf* disk file, which contains the timing for instance *u1* of design *MULT16*. The timing is annotated on the design *MY_DESIGN*. In this case, load delay is included in the net delays.

```
prompt> current_design MY_DESIGN
```

```
prompt> read_sdf -load_delay net -path u1 mult16_u1.sdf
```

The following example reads minimum and maximum timing information from two separate SDF files and annotates the current design with delays corresponding to minimum and maximum operating conditions, respectively:

```
prompt> read_sdf -min_file min.sdf -max_file max.sdf
```

SEE ALSO

- `change_names(2)`
- `remove_annotated_check(2)`
- `remove_annotated_delay(2)`
- `report_annotated_check(2)`
- `report_annotated_delay(2)`
- `reset_design(2)`
- `set_annotated_check(2)`
- `set_annotated_delay(2)`
- `write_sdf(2)`

read_sverilog

Reads in one or more design or library files in SystemVerilog format.

SYNTAX

```
string read_sverilog  
    file_names  
    [-netlist]  
    [-rtl]
```

Data Types

file_names list

ARGUMENTS

file_names

Specifies the names of one or more files to be read.

DESCRIPTION

The **read_sverilog** command reads design information from SystemVerilog files into dc_shell.

This command is derived from **read_file -format sverilog**. For more information, see the man page for the **read_file** command.

SEE ALSO

read_file(2)

read_test_model

Reads a test model file.

SYNTAX

```
list read_test_model  
  [-format ddc | ctl]  
  [-design design_name]  
  model_files
```

Data Types

design_name string
model_files list

ARGUMENTS

-format ddc | ctl

Specifies the format of the test model. A test model can be stored in a design .ddc file (default) or as ASCII text in a Core Test Language (CTL) file.

By default, the model is assumed to be in .ddc format.

-design *design_name*

Specifies the name of the design to which to attach the test model. This is meaningful only for CTL test models. By default, the model is attached to the current design.

model_files

Specifies the names of the test model files to read. This argument is required. When you specify the format as **ctl**, you can specify only one file.

DESCRIPTION

The **read_test_model** command reads test model files.

The test model files represent the test behavior of designs. The attributes that are represented include all DFT signals and their purpose, the test timing, and the sequence of events needed to shift test vectors in and out of the design. For more information about attributes that may be represented, see the *DFT Compiler Scan User Guide* and the *DFTMAX Compression User Guide*.

When the format is specified as **ddc**, you can specify multiple model files to read. Each model of each .ddc file is read in. Since .ddc models include the designation of the designs they model, the **-design** option is ignored.

The **read_test_model** command reads in only the test model and the interface definition of a design, and ignores any other information. Since the gate implementation of a design is not loaded in the database, significant memory savings can be achieved. For a .ddc file containing only design interface definitions and test models, the **read_file** and **read_test_model** commands are synonymous.

When the format is specified as **ctl**, a single test model is read from a CTL (ASCII) file and is associated with the current design, or the design specified through the **-design** option. The CTL format is used for test models generated outside of a Synopsys environment.

A successful read of a model overwrites any test model information in memory.

Test models are stored on disk using the **write_test_model** or the **write** command. Transfer test models from one session to another either by storing full designs, or by storing only test models. Unless there is a requirement to hide a design's implementation, it is best to store the full designs, since it reduces the risk of confusion about the contents of the .ddc files.

EXAMPLES

The following command sequence stores full designs to disk. It facilitates external data management by having a single .ddc file for a design.

```
prompt> write -format ddc MyDesign -out MyDesign.ddc
```

```
prompt> remove_design MyDesign
```

```
prompt> read_test_model MyDesign.ddc
```

The following command sequence stores only the test model to a disk file. It requires the separate management of the implementation of the design.

```
prompt> write_test_model MyDesign -out MyDesign.ddc
```

```
prompt> remove_design MyDesign
```

```
prompt> read_file MyDesign.ddc
```

The last **read_file** command above is equivalent to a **read_test_model** command, because the *MyDesign.ddc* file contains only the design interface and the test model.

SEE ALSO

- `list_test_models(2)`
- `remove_test_model(2)`
- `report_test_model(2)`
- `write_test_model(2)`

read_test_protocol

Reads a STIL test protocol file into memory.

SYNTAX

```
status read_test_protocol
[-verbose]
[-test_mode test_mode_name]
[-overwrite]
[-section section_name]
input_file_name
```

Data Types

```
test_mode_name  string
section_name   string
input_file_name string
```

ARGUMENTS

-verbose

Specifies that verbose output is displayed when set to **true**.

-test_mode *test_mode_name*

Specifies a test mode name to read and store a test protocol. If not specified, the current test mode is used by default. This option is only supported when updating the post-DFT test protocol.

-overwrite

Specifies to overwrite timing, procedures, or macros if they already exist in memory when reading in a new test protocol. For example, if there is already a protocol in the memory with timing specified, and you read in a new protocol with different timing, this option specifies to overwrite the existing timing with the new timing.

-section *section_name*

Specifies a section name of the protocol. With this option, the **read_test_protocol** command reads only the specified section and ignores others. The only valid parameter is **test_setup**, which describes the initialization sequence.

input_file_name

Specifies the name of the test protocol file in STIL format to read. The default file name is *current_design_name.spf*.

DESCRIPTION

This command reads a test protocol file into memory. Reading a test protocol file defines the scan test protocol for the current design.

The scan test protocol provides a means to formally define the process by which a design is tested, such as the sequence of scan-in vectors, parallel vectors, and scan-out vectors performed for each test pattern. The protocol defined for a design is used to drive scan-test design rule checking.

The format of the test protocol file is STIL. See the DFT Compiler, DFTMAX, and TetraMAX user guides for details of the protocol file format.

The **read_test_protocol** command enables you to define an arbitrary test protocol for a design by reading in a text file describing the protocol. Use **write_test_protocol** to write out an existing protocol for a design in the same text format.

Normally, this command removes any protocol that may be present in memory through the previous use of the **read_test_protocol** or **create_test_protocol** commands.

When the **read_test_protocol -section test_setup** command is executed before pre-DFT DRC has been performed, the **test_setup** section from the specified protocol file is read in and used in place of the default setup protocol created by DFT Compiler. The provided **test_setup** section must include all preconditioned signals defined by the **set_dft_signal** command, such as clock, set, reset, and constant signals. Any STIL **Loop** constructs are retained as-is.

To modify the default setup protocol used by pre-DFT DRC, you can write out the default test protocol, modify the **test_setup** section, then read it back in; see the EXAMPLES section for more information.

When the **read_test_protocol -section test_setup** command is executed after DFT insertion has been performed, the **test_setup** section of the in-memory test protocol created by DFT Compiler is merged with the **test_setup** section from the specified protocol file. This merged in-memory protocol is used for any subsequent DRC. Any STIL **Loop** constructs are unrolled during this merging process. If multiple test modes were created by DFT Compiler, you can update each one using the **-test_mode** option of this command, or the **current_test_mode** command.

If the test protocol file name is not specified for this command, the default is *current_design_name.spf*.

With the STIL protocol format, you cannot distinguish between a clock signal and an asynchronous signal. You must define asynchronous signals with the **set_dft_signal** command before reading your custom STIL protocol.

EXAMPLES

The following example reads a test protocol from the **newUPC1.spf** file:

```
prompt> read_test_protocol newUPC1.spf
Reading test protocol file 'newUPC1.spf'...
```

The following example reads a test protocol for the **UPC1** design from the **UPC1.spf** default file:

```
prompt> read_test_protocol
Reading test protocol file 'UPC1.spf'...
```

The following example defines the asynchronous signal **rstn** for design **UPC1** and reads a test protocol from the file **UPC1.spf**:

```
prompt> set_dft_signal -type Reset -active_state 0 -port rstn

prompt> read_test_protocol UPC1.spf
```

The following example reads a **test_setup** section of the protocol. For example, assume that the **init_sequence.spf** file has the following **test_setup** vector sequence:

```
MacroDefs {
  "test_setup" {
    W "_default_WFT_";
    V { "A" = 0; "B" = 0; "CP" = 0; }
    V { "A" = 1; "B" = 1; "CP" = P; }
    V { "A" = 0; "B" = 0; "CP" = 1; "AA" = 1; "BB" = 1; }
  }
}
```

The following command reads in the initialization sequence and ignores the rest of the protocol file:

```
prompt> read_test_protocol -section test_setup init_sequence.spf
```

The following example creates a default test protocol, modifies it, and reads the modified **test_setup** back in:

```
prompt> create_test_protocol
```

```
prompt> write_test_protocol -output test_setup_mod.spf
```

(manually modify **test_setup** in file,
keeping pre-conditioned signals as needed)

```
prompt> read_test_protocol -section test_setup test_setup_mod.spf
```

The following example updates the post-DFT test setup protocol for two test modes:

```
prompt> insert_dft
```

```
prompt> read_test_protocol -section test_setup \
  -test_mode Internal_scan postDFT.spf
```

```
prompt> read_test_protocol -section test_setup \
  -test_mode ScanCompression_mode postDFT.spf
```

```
prompt> current_test_mode Internal_scan
```

```
prompt> dft_drc
```

```
prompt> current_test_mode ScanCompression_mode
```

```
prompt> dft_drc
```

SEE ALSO

```
create_test_protocol(2)
dft_drc(2)
remove_test_protocol(2)
set_dft_signal(2)
write_test_protocol(2)
current_design(2)
```

read_verilog

Reads in one or more design or library files in Verilog format.

SYNTAX

```
status read_verilog  
[-netlist | -rtl]  
verilog_files
```

Data Types

verilog_files list

ARGUMENTS

-netlist

Specifies that the design being read is a structural or gate-level design. This option invokes a netlist reader optimized for gate-level designs. If any of the specified files contain RTL-level constructs, an error is issued. Use of the **-netlist** option might enable the tool to read larger gate-level designs than would otherwise be the case.

This option is mutually exclusive with the **-rtl** option.

-rtl

Specifies that the design being read is an RTL design. This option invokes the Verilog RTL reader directly without attempting to determine if the specified files are gate-level netlists.

This option is mutually exclusive with the **-netlist** option.

verilog_files

Specifies the name of one or more Verilog files to be read.

DESCRIPTION

The **read_verilog** command reads design information from Verilog files into memory.

If the automatic mode is on, the netlist reader (read with the **-netlist** option) is automatically invoked. If the input contains RTL constructs, the system automatically invokes the RTL reader (read with **-rtl** option).

This command is derived from **read_file -format verilog**. For more information, see the man page for the **read_file** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

`read_file(2)`

read_vhdl

Reads in one or more designs or library files in VHDL format.

SYNTAX

```
string read_vhdl  
  [-netlist]  
  file_names
```

Data Types

file_names list

ARGUMENTS

-netlist

Specifies that the design being read is a structural or gate-level design. This option invokes a netlist reader optimized for gate-level designs. If any of the given files contains RTL-level constructs, the command issues an error. Using the **-netlist** option enables the command to read large gate-level designs.

file_names

Specifies the names of the design files to be read.

DESCRIPTION

This command reads design information from VHDL files into the appropriate shell. It is derived from the **read_file** command for use with VHDL files. For more information, see the man page of the **read_file** command.

SEE ALSO

read_file(2)

rebuild_mw_lib

Rebuilds the Milkyway library.

SYNTAX

status_value **rebuild_mw_lib**
libName

Data Types

libName string

ARGUMENTS

libName

Specifies the Milkyway library to be rebuilt. The value of *mw_lib* should be a valid library name. The library must be closed for this command to work.

DESCRIPTION

This command rebuilds a Milkyway library by scanning all designs in the associated library directory.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example rebuilds the current Milkyway library:

```
prompt> rebuild_mw_lib
Scanning library...
place.CEL;5
place.CEL;4
place.CEL;3
place.CEL;2
```

```
place.CEL;1
place.EXP;1
place.NETL;1
place.PARA;1
Rebuilding library...
A total of 8 items have been rebuilt. (0 removed, 0 new added)
1
```

SEE ALSO

```
close_mw_lib(2)
copy_mw_lib(2)
create_mw_lib(2)
current_mw_lib(2)
open_mw_lib(2)
rename_mw_lib(2)
report_mw_lib(2)
```

redirect

Redirects the output of a command to a file.

SYNTAX

```
string redirect  
  [-append] [-tee] [-file  
  | -variable  
  | -channel] [-compress]  
  [-bg]  
  [-max_cores number_of_cores]  
  target  
  {command_string}
```

Data Types

```
target      string  
command_string  string
```

ARGUMENTS

-append

Appends the output to *target*.

-tee

Like the unix command of the same name, sends output to the current output channel as well as to the *target*.

-file

Indicates that *target* is a file name, and redirection is to that file. This is the default. It is exclusive of *-variable* and *-channel*.

-variable

Indicates that *target* is a variable name, and redirection is to that Tcl variable. It is exclusive of *-file* and *-channel*.

-channel

Indicates tha *target* is a Tcl channel, and redirection is to that channel. It is exclusive of *-variable* and *-file*.

-compress

Compress when writing to file. If redirecting to a file, then this option specifies that the output should be compressed as it is written. The output file is in a format recognizable by "gzip -d". You cannot specify this option with *-append*.

-bg

Indicates that the redirected command executes in the background and in parallel with other foreground commands downstream. The `dc_shell` returns immediately to execute the next command after this redirect command as long as there is only two redirect -bg executed at one point. If there are already two redirect -bg commands running, the next redirect -bg will be blocking. Also exit or quit command blocks until all the background commands started with the -bg option are finished. Note the -bg option must be used together with the -file option and -max_cores option. No other options are allowed. Redirect command returns the PID of the background process when this option is specified.

Only limited set of `dc_shell` commands are supported within redirect -bg. When an unsupported command is executed within redirect -bg, an error message will be printed. To report all the supported commands, you can run the command **list_commands -bg**.

-max_cores number_of_cores

Specifies the maximum number of CPU cores that can be used by multicore commands run within redirect -bg

target

Indicates the target of the output redirection. This is either a filename, Tcl variable, or Tcl channel depending on the command arguments.

command_string

The command to execute. Intermediate output from this command, as well as the result of the command, will be redirected to *target*. The *command_string* should be rigidly quoted with curly braces.

DESCRIPTION

The **redirect** command performs the same function as the traditional unix-style redirection operators > and >>. The *command_string* must be rigidly quoted (that is, enclosed in curly braces) in order for the operation to succeed.

Output is redirected to a file by default. Output can be redirected to a Tcl variable by using the *-variable* option, or to a Tcl channel by using the *-channel* option.

Output can be sent to the current output device as well as the redirect target by using the *-tee* option. See the examples section for an example.

The result of a **redirect** command which does not generate a Tcl error is the empty string. Screen output occurs only if errors occurred during execution of the *command_string* (other than opening the redirect file). When errors occur, a summary message is printed. See the examples.

Although the result of a successful **redirect** command is the empty string, it is still possible to get and use the result of the command that you redirected. Construct a **set** command in which you set a variable to the result of your command. Then, redirect the **set** command. The variable holds the result of your command. See the examples.

The **redirect** command is much more flexible than traditional unix redirection operators. With **redirect**, you can redirect multiple commands or an entire script. See the examples for an example of how to construct such a command.

Note that the builtin Tcl command **puts** does not respond to output redirection of any kind. Use the builtin **echo** command instead.

This -bg option is introduced to enable parallel processing on multicore hardware for performance reasons. It is also very important to understand that the `background_execute` is targeted for post-update reporting, analysis and design export commands. If any of the command executed in the background causes changes to netlist or any other data, those changes will not be brought back to the foreground job that is running. Also, please be aware that use of commands modifying the netlist or other data can cause memory blow up in the child process.

If you make use of the `change_names` command within redirect -bg, then you need to stop recording formality setup information

before running the `redirect -bg` command. And use a new svf file within `redirect -bg`. Both the svf files should be given for formal verification.

The maximum number cores used in the main session will be reduced by the max cores of the `redirect -bg` job(s) that is running. You can make use of `parallel_execute` within `background_execute`

EXAMPLES

In the following example, the output of the `plus` procedure is redirected. The echoed string and the result of the `plus` operation is in the output file. Notice that the result was not echoed to the screen.

```
prompt> proc plus {a b} {echo "In plus" ; return [expr $a + $b]}
prompt> redirect p.out {plus 12 13}
prompt> exec cat p.out
In plus
25
```

In this example, a typo in the command created an error condition. The error message indicates that you can use **error_info** to trace the error, but you should first check the output file.

```
prompt> redirect p.out {plus2 12 13}
Error: Errors detected during redirect
      Use error_info for more info. (CMD-013)
prompt> exec cat p.out
Error: unknown command 'plus2' (CMD-005)
```

In this example, we explore the usage of results from redirected commands. Since the result of **redirect** for a command which does not generate a Tcl error is the empty string, use the **set** command to trap the result of the command. For example, assume that there is a command to read a file which has a result of "1" if it succeeds, and "0" if it fails. If you redirect only the command, there is no way to know if it succeeded.

```
redirect p.out { read_a_file "a.txt" }
# Now what? How can I redirect and use the result?
```

But if you set a variable to the result, then it is possible to use that result in a conditional expression, etc.

```
redirect p.out { set rres [read_a_file "a.txt"] }
if { $rres == 1 } {
  echo "Read ok!"
}
```

The **redirect** command is not limited to redirection of a single command. You can redirect entire blocks of a script with a single **redirect** command. This simple example with **echo** demonstrates this feature:

```
prompt> redirect p.out {
?   echo -n "Hello "
```

```
?    echo "world"
?    }
prompt> exec cat p.out
Hello world
prompt>
```

The **redirect** command allows you to tee output to the previous output device and also to redirect output to a variable. This simple example with **echo** demonstrates these features:

```
prompt> set y "This is "
This is
prompt> redirect -tee x.out {
    echo XXX
    redirect -variable y -append {
    echo YYY
    redirect -tee -variable z {
    echo ZZZ
    }
    }
}
XXX
prompt> exec cat x.out
XXX
prompt> echo $y
This is YYY
ZZZ

prompt> echo $z
ZZZ
```

The following example demonstrates how **redirect** command with **-bg** option can be used to run a tcl procedure and a tcl script. The tcl procedure and tcl script can contain any of the dc_shell command supported by **-bg** option.

```
prompt> set_host_options -max_cores 4
prompt> redirect -bg -max_cores 1 -file x.out {MyTclProc}
prompt> redirect -bg -max_cores 1 -file y.out {source y.tcl}
prompt> echo $z
ZZZ
```

The following example demonstrates how **set_svf** command can be used within **redirect -bg**. If the reference design is from foo.v, then to verify the netlist after **redirect -bg**, the svf files compile.svf and redirect.svf should be used. And to verify the netlist after **compile_ultra -incr**, the svf files compile.svf and compile_incr.svf should be used.

```
prompt> set_host_options -max_cores 4
prompt> set_svf compile.svf
prompt> read_verilog foo.v
prompt> read_sdc foo.sdc
prompt> compile_ultra -spg
prompt> set_svf -off
prompt> redirect -bg -max_cores 1 -file x.out {set_svf redirect.svf; change_names -rules verilog -hier; ...}
```

```
prompt> set_svf compile_incr.svf  
prompt>compile_ultra -spg -incr  
ZZZ
```

SEE ALSO

- echo(2)
- error_info(2)
- set(2)
- report_background_jobs(2)
- parallel_execute(2)
- list_commands(2)

remove_annotated_check

Removes annotated timing-check information.

SYNTAX

```
status remove_annotated_check  
-all | -from from_list | -to to_list  
[-rise | -fall]  
[-clock rise | fall]  
[-setup]  
[-hold]  
[-recovery]  
[-removal]  
[-nochange_low]  
[-nochange_high]
```

Data Types

```
from_list  list  
to_list    list
```

ARGUMENTS

-all

Removes all annotated checks in the current design. This includes rise and fall values for setup, hold, recovery, removal, and nochange pin-to-pin timing checks. You must specify either **-all**, **-from**, or **-to**; if you specify **-all**, you cannot specify any other arguments or options.

-from *from_list*

Specifies a list of leaf cell clock pins that are the startpoints of the timing arcs from which annotated checks are to be removed. You must specify either **-all**, **-from**, or **-to**; if you specify **-all**, you cannot specify any other arguments or options.

-to *to_list*

Specifies a list of leaf cell data pins that are the endpoints of the timing arcs from which annotated checks are to be removed. You must specify either **-all**, **-from**, or **-to**; if you specify **-all**, you cannot specify any other arguments or options.

-rise | -fall

Specifies whether the check to be removed is for data rise or fall transition. If you do not specify either **-rise** or **-fall**, both values are removed.

-clock rise | fall

Specifies whether the check to remove is for clock rising or falling. By default, checks for both clock rise and fall are removed.

-setup

Removes only setup timing-check information.

-hold

Removes only hold timing-check information.

-recovery

Removes only recovery timing-check information.

-removal

Removes only removal timing-check information.

-nochange_low

Removes only the nochange timing check against data low information.

-nochange_high

Removes only the nochange timing check against data high information.

DESCRIPTION

This command removes annotated timing checks between the specified pins. Data rise and fall and clock rise and fall annotated checks are removed by default.

You can use the **remove_annotated_check** command for pins at lower levels of the design hierarchy. These pins are specified as "INSTANCE1/INSTANCE2/PIN_NAME."

If the design is not already linked, **remove_annotated_check** links it automatically.

The **remove_annotated_check** command removes annotated timing checks set by the **set_annotated_check** command, and also removes setup, hold, recovery, removal, or nochange annotated timing checks set by the **read_sdf** command. The **reset_design** command removes annotated timing checks in addition to other information.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes an annotated setup check between pins u1/u2/CP and u1/u2/D:

```
prompt> remove_annotated_check -setup -from u1/u2/CP -to u1/u2/D
```

The following example removes all annotated timing checks on the current design:

```
prompt> remove_annotated_check -all
```

SEE ALSO

current_design(2)
link(2)
read_sdf(2)
report_annotated_check(2)
reset_design(2)
set_annotated_check(2)

remove_annotated_delay

Removes the annotated delay between two pins.

SYNTAX

```
status remove_annotated_delay
  -all
  | -cell_all
  | -net_all
  | -non_clock_cell_all
  | -non_clock_net_all
  | -from from_list
  | -to to_list
```

Data Types

```
from_list  list
to_list    list
```

ARGUMENTS

-all

Removes all net- and cell-annotated delays in the current design. You must specify either **-all**, **-from**, or **-to**; if you specify **-all**, you cannot specify any other arguments.

-cell_all

Removes all cell-annotated delays in the current design. You can specify **-cell_all** and **-net_all** together to remove all cell- and net-annotated delays.

-net_all

Removes all net-annotated delays in the current design. You can specify **-cell_all** and **-net_all** together to remove all cell- and net-annotated delays.

-non_clock_cell_all

Removes all cell-annotated delays in the current design except those on the clock network. You can specify **-non_clock_cell_all** and **-non_clock_net_all** together to remove all cell- and net-annotated delays not on the clock network.

-non_clock_net_all

Removes all net-annotated delays in the current design except those on the clock network. You can specify **-non_clock_cell_all** and **-non_clock_net_all** together to remove all cell- and net-annotated delays not on the clock network.

-from *from_list*

Specifies a list of leaf-cell pins or top-level ports that are the startpoints of the timing arcs from which to remove annotated delays. You must specify either **-all**, **-from**, or **-to**; if you specify **-all**, you cannot specify any other arguments.

-to *to_list*

Specifies a list of leaf-cell pins or top-level ports that are the endpoints of the timing arcs from which to remove annotated delays. You must specify either **-all**, **-from**, or **-to**; if you specify **-all**, you cannot specify any other arguments.

DESCRIPTION

This command removes annotated delays between the specified pins, which must be on the same net or on the same cell. Both rise and fall annotated delays are removed. There must be a timing arc between the pins in *from_list* and the pins in *to_list* or no annotated delay is removed.

You can use **remove_annotated_delay** for pins at lower levels of the design hierarchy. These pins are specified as "INSTANCE1/INSTANCE2/PIN_NAME."

If the current design is hierarchical, you must link the design with the **link -all** command before using **remove_annotated_delay**.

To annotate net delay values between pins in a design, use **set_annotated_delay -net** command. To annotate cell delay values between pins in a design, use the **set_annotated_delay -cell** command. You must use **-from** and **-to** in the same manner you used them to set annotated values. For example, use **remove_annotated_delay -from** to remove an annotated delay set with **set_annotated_delay -from**. To remove an annotated delay set with **set_annotated_delay -to**, use **remove_annotated_delay -to**. To remove an annotated delay set with **set_annotated_delay -from -to**, use **remove_annotated_delay -from -to**.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example first shows the annotation of the delay to remove. The second command removes the annotated net delay between output pin Z of cell instance U1/U2/U3 and input pin A of cell instance U6:

```
prompt> set_annotated_delay -net <value> -from U1/U2/U3/Z -to U6/A
prompt> remove_annotated_delay -from U1/U2/U3/Z -to U6/A
```

This example first shows the annotation of the delay to remove. The second command removes the annotated, lumped net delay between output pin Z of cell instance U1/U2/U3 and all fanout pins on that net:

```
prompt> set_annotated_delay -net <value> -from U1/U2/U3/Z
prompt> remove_annotated_delay -from U1/U2/U3/Z
```

The following example removes all annotated net and cell delays from the current design:

```
prompt> remove_annotated_delay -all
```

SEE ALSO

current_design(2)
link(2)
report_timing(2)
reset_design(2)
set_annotated_delay(2)

remove_annotated_transition

Removes the annotated transition at a pin.

SYNTAX

```
status remove_annotated_transition  
-all
```

ARGUMENTS

-all

Removes the annotated transition time from every pin in the current design. You must specify either **-all** or **-at**; if you specify **-all**, you cannot specify any other arguments.

DESCRIPTION

This command removes annotated transitions at the specified pins. Both rise and fall annotated transitions are removed.

To annotate net-transition values at pins in a design, use the **set_annotated_transition** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes the annotated transition at every pin in the design:

```
prompt> remove_annotated_transition -all
```

The following example first shows the annotation of the transition to be removed. The second command removes the annotated transition at the output pin Z of cell instance U1/U2/U3:

```
prompt> set_annotated_transition value U1/U2/U3/Z  
prompt> remove_annotated_transition -at U1/U2/U3/Z
```

SEE ALSO

current_design(2)
link(2)
report_timing(2)
reset_design(2)
set_annotated_transition(2)

remove_annotations

Removes all annotated information on the design.

SYNTAX

status **remove_annotations**

ARGUMENTS

The **remove_annotations** command has no arguments.

DESCRIPTION

The **remove_annotations** command removes all annotations on the design, including delay, transition, resistance, capacitance, and check. These include the data set by the **set_annotated_check**, **set_annotated_delay**, **set_annotated_transition**, **set_resistance**, and **set_load** commands.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes all annotated information set by previous commands:

```
prompt> set_annotated_check 0.25 -from m2/st_reg[0]/CP \
      -to m2/st_reg[0]/D -setup
```

```
Warning: There is no 'setup_falling' timing arc between pins
'm2/st_reg[0]/CP' and 'm2/st_reg[0]/D'. (OPT-815)
```

```
1
```

```
prompt> set_annotated_delay -cell 0.03 -from m1/U18/A -to m1/U18/Z
```

```
Information: Annotated 'cell' delays are assumed to include load delay. (UID-282)
```

```
1
```

```
prompt> set_annotated_transition 0.07 m2/st_reg[0]/D
```

1

prompt> **remove_annotations**

1

SEE ALSO

current_design(2)
link(2)
remove_annotated_check(2)
remove_annotated_delay(2)
remove_annotated_transition(2)
remove_sdc(2)
report_timing(2)
reset_design(2)
set_load(2)
set_resistance(2)

remove_attribute

Removes an attribute from the specified objects.

SYNTAX

```
collection remove_attribute
  [-bus]
  [-quiet]
  object_list
  attribute_name
```

Data Types

```
object_list    list
attribute_name string
```

ARGUMENTS

-bus

Indicates to remove the attribute from the bus as a whole, and not from each individual bus member.

-quiet

Turns off the warning messages that would otherwise be issued if the attribute or objects are not found.

object_list

Specifies a list of objects from which the attribute is to be removed.

attribute_name

Specifies the name of the attribute to be removed.

DESCRIPTION

This command removes the specified attribute from the specified objects. For a complete list of attributes, see the **attributes** man page.

This command returns the list of objects from which the attribute is removed. A returned empty string indicates that no object has been removed.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information if there is no scenario dependency for the attributes. However, for attributes such as **max_leakage_power**, which are scenario dependent, this command uses information from the current scenario only.

EXAMPLES

This command removes the **output_not_used** attribute on a port:

```
prompt> remove_attribute [get_ports OUT1] output_not_used  
{OUT1}
```

In the following example, a warning message is issued because the cell named *U9* cannot be found:

```
prompt> remove_attribute [get_cells {U1 U9}] dont_touch  
Warning: Can't find object 'U9' in design 'example'  
{U1}
```

Using the **-quiet** option with the previous example inhibits the warning messages:

```
prompt> remove_attribute -quiet [get_cells {U1 U9}] dont_touch  
{U1}
```

In this example, the **dont_use** attribute of a library cell is removed:

```
prompt> remove_attribute [get_lib_cells tech_lib/GATE] dont_use  
{tech_lib/GATE}
```

In the following example, the specified attribute cannot be found:

```
prompt> remove_attribute [get_cells *] arrival
```

SEE ALSO

- collections(2)
- define_user_attribute(2)
- get_attribute(2)
- list_attributes(2)
- reset_design(2)
- set_attribute(2)
- attributes(3)

remove_auto_path_groups

Removes path groups for the current design.

SYNTAX

```
string remove_auto_path_groups  
  [-prefix prefix]  
  [-file file_name]  
  [-verbose]  
  [-user_path_groups_file user_file_name]  
  [-skip]
```

Data Types

```
prefix      string  
file_name   string  
user_file_name string
```

ARGUMENTS

-prefix *prefix*

Specifies the prefix of the path group names. The default is `synopsys_pg_`.

-file *file_name*

Writes **remove_path_group** commands to the specified file.

-verbose

Activates the verbose mode.

-user_path_groups_file *user_file_name*

Specifies the file from which user path groups are restored. The default is `auto_path_groups.user_path_groups.tcl`.

DESCRIPTION

Removes path groups created by the **create_auto_path_groups** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

For examples, see the **create_auto_path_groups** command man page.

SEE ALSO

`create_auto_path_groups(2)`

remove_boundary_cell

Removes the boundary-cell configuration for the specified ports or core cells.

SYNTAX

```
status remove_boundary_cell  
-class core_wrapper | shadow_wrapper | bsd  
-ports port_list | -core core_cell_list  
[-function function_type]
```

Data Types

```
port_list      list  
core_cell_list string  
function_type string
```

ARGUMENTS

-class *core_wrapper* | *shadow_wrapper* | *bsd*

Specifies the name of the class for which the boundary-cell configuration is to be removed.

Valid values for this option are **core_wrapper**, **shadow_wrapper**, and **bsd**.

This option is required.

-ports *port_list*

Specifies the list of ports for which the boundary-cell configuration is to be removed.

There is no default value for this option. You must use either the **-ports** option or the **-core** option.

-core *core_cell_list*

Specifies the list of core cells for which the boundary cell configuration is to be removed.

When the *core_list* is not empty, the *port_list* specifies the ports of these core cells.

There is no default value for this option. You must use either the **-ports** option or the **-core** option.

-function *function_type*

Specifies the function of the specified ports for which the boundary cell configuration is to be removed.

Valid values for this option are as follows:

input

output
control
bidir
observe
input_inverted
output_inverted
bidir_inverted

There is no default value for this option.

DESCRIPTION

The **remove_boundary_cell** command removes the boundary-cell configuration on the specified core cells or ports for the specified class. The core cells or ports must exist in the current design.

EXAMPLES

The following command sets the boundary-cell configuration on *port1*:

```
prompt> set_boundary_cell -class core_wrapper -type WC_D1 \  
-safe_state none -port_list port1  
1
```

The following command sets the boundary-cell information on *port2*:

```
prompt> set_boundary_cell -class core_wrapper -type WC_S1_S \  
-safe_state 0 -port_list port2  
1
```

The following command removes the boundary cell configuration for the core wrapper class on the *port1* and *port2* ports:

```
prompt> remove_boundary_cell -class core_wrapper \  
-port_list {port1, port2}  
1
```

SEE ALSO

insert_dft(2)
preview_dft(2)
set_boundary_cell(2)
set_bsd_configuration(2)
set_wrapper_configuration(2)

remove_boundary_cell_io

Removes the boundary-cell I/O specifications from the current boundary-scan design.

SYNTAX

```
status remove_boundary_cell_io
  [-all]
  [-cell cell_list]
```

Data Types

cell_list string

ARGUMENTS

-all

Removes all boundary-cell I/O specifications from the design.

-cell *cell_list*

Removes all user-defined boundary I/O cells specified in the current design.

DESCRIPTION

The **remove_boundary_cell_io** command removes the boundary-cell I/O specifications from the current design. If you invoke **remove_boundary_cell_io** with no options, the command is ignored. You receive an informational message describing this action.

EXAMPLES

The following example removes the boundary-cell I/O specification from the design:

```
prompt> set_boundary_cell_io -type boundary \
-cell what_testcase_bs_core_inst/bsr_gpio_3_inst \
-access {out_pi what_testcase_bs_core_inst/U21/Z \
out_po what_testcase_bs_core_inst/bsr_gpio_3_inst/U11/Z}

prompt> remove_boundary_cell_io -cell {what_testcase_bs_core_inst/bsr_gpio_3_inst}
```

The following example removes all boundary cell I/O specifications from the design:

```
prompt> set_boundary_cell_io -type boundary \  
-cell what_testcase_bs_core_inst/bsr_gpio_3_inst \  
-access {out_pi what_testcase_bs_core_inst/what_testcase_core_inst/U21/Z \  
out_po what_testcase_bs_core_inst/bsr_gpio_3_inst/U11/Z}  
  
prompt> set_boundary_cell_io -type boundary \  
-cell what_testcase_bs_core_inst/bsr_gpio_0_inst \  
-access {out_pi what_testcase_bs_core_inst/what_testcase_core_inst/U19/Z \  
out_po what_testcase_bs_core_inst/bsr_gpio_0_inst/U22/Z}  
  
prompt> remove_boundary_cell_io -all
```

SEE ALSO

report_boundary_cell_io(2)
set_boundary_cell_io(2)

remove_bounds

Removes bounds from the current design.

SYNTAX

```
status remove_bounds
  [-verbose]
  [-all]
  [-name bound_name_list]
  objects
```

Data Types

<i>bound_name_list</i>	list
<i>objects</i>	string

ARGUMENTS

-verbose

Prints information about what is being deleted.

-all

Removes all bounds from the current design. This command removes only bounds that have been created using the **create_bounds** command. Bounds created in an input PDEF file cannot be removed.

-name *bound_name_list*

Specifies to remove bounds with the given names.

objects

Specifies bound objects to be removed.

DESCRIPTION

This command removes bounds constraints from the current design. If you specify the **-all** option, all bounds are removed from the design, with the exception of those specified in an input PDEF file. If you give a list of bound names, the bounds with the specified names are removed. The ID is displayed every time a bound is created. You can use the **report_bounds** command to view the ID.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the bounds named *temp_0* and *temp_1*.

```
prompt> remove_bounds -name {temp_0 temp_1}
```

SEE ALSO

- create_bounds(2)
- report_bounds(2)
- set_cell_location(2)
- update_bounds(2)

remove_bsd_compliance

Removes the compliance ports specifications from the current boundary-scan design.

SYNTAX

```
status remove_bsd_compliance  
  [-all]  
  [-name pattern_name]
```

Data Types

pattern_name string

ARGUMENTS

-all

Removes all compliance port specifications from the design.

-name *pattern_name*

Removes all user-defined pattern names specified if present in the current design.

DESCRIPTION

The **remove_bsd_compliance** command removes the compliance ports specifications from the current design. If you invoke **remove_bsd_compliance** with no options, the command is ignored. You receive an informational message describing this action.

EXAMPLES

The following example sets and then removes the compliance port specification from the design:

```
prompt> set_bsd_compliance -name pat1 -pattern {my_bidi 0 my_eni 1}
```

```
prompt> remove_bsd_compliance -name pat1
```

The following example sets and then removes all compliance ports specifications from the design:

```
prompt> set_bsd_compliance -name pat1 -pattern {my_bidi 0 my_eni 1}
```

```
prompt> set_bsd_compliance -name pat2 -pattern {my_ini 0}
```

```
prompt> remove_bsd_compliance -all
```

SEE ALSO

report_bsd_compliance(2)

set_bsd_compliance(2)

remove_bsd_instruction

Removes boundary-scan instructions from the instruction list to be used by **insert_dft** for the current design.

SYNTAX

```
status remove_bsd_instruction  
      instruction_list
```

Data Types

instruction_list list

ARGUMENTS

instruction_list

Specifies a comma-separated list containing one or more boundary-scan instructions to remove from the set of boundary-scan instructions for the current design.

DESCRIPTION

The **remove_bsd_instruction** command specifies one or more boundary-scan instructions to be removed from the set of boundary-scan instructions to be implemented by **insert_dft** for the current design. You can specify only the BYPASS, EXTEST, SAMPLE, IDCODE, CLAMP, and HIGHZ instructions using this command.

EXAMPLE

The following example removes IDCODE and CLAMP from the list of boundary scan instructions to be used by the **insert_dft** command:

```
prompt> remove_bsd_instruction {IDCODE, CLAMP}
```

SEE ALSO

insert_dft(2)
set_bsd_instruction(2)

remove_bsd_linkage_port

Removes the linkage-ports specifications from the current boundary-scan design.

SYNTAX

```
status remove_bsd_linkage_port  
  [-all]  
  [-port_list port_list]
```

Data Types

port_list string

ARGUMENTS

-all

Removes all linkage-ports specifications from the design.

-port_list *port_list*

Removes all user-defined linkage ports specified in the current design.

DESCRIPTION

The **remove_bsd_linkage_port** command removes the linkage-ports specifications from the current design. If you invoke **remove_bsd_linkage_port** with no options, the command is ignored and you receive an informational message.

EXAMPLES

The following example removes the linkage port specification from the design:

```
prompt> set_bsd_linkage_port -port_list {my_bidi my_ini out1}
```

```
prompt> remove_bsd_linkage_port -port_list {my_bidi out1}
```

The following example removes all linkage ports specifications from the design:

```
prompt> set_bsd_linkage_port -port_list {my_bidi my_ini out1}
```

```
prompt> remove_bsd_linkage_port -all
```

SEE ALSO

report_bsd_linkage_port(2)

set_bsd_linkage_port(2)

remove_bsd_power_up_reset

Removes the power-up reset port specifications from the current boundary-scan design.

SYNTAX

status **remove_bsd_power_up_reset**

ARGUMENTS

The **remove_bsd_power_up_reset** command has no arguments.

DESCRIPTION

The **remove_bsd_power_up_reset** command removes the power-up reset port specifications from the current design.

EXAMPLES

The following example removes the power-up reset port specifications from the design:

```
prompt> set_bsd_power_up_reset -reset_pin_name in0 -active high \  
         -delay 500 -cell_name U15
```

```
prompt> remove_bsd_power_up_reset
```

SEE ALSO

[report_bsd_power_up_reset\(2\)](#)
[set_bsd_power_up_reset\(2\)](#)

remove_buffer

Removes the buffer cells at a specified driver pin or net on a mapped design.

SYNTAX

```
status remove_buffer  
-from start_point  
-net net_list  
[-to end_point_list]  
[-level number_of_levels]  
cell_list
```

Data Types

```
start_point    list  
net_list       list  
end_point_list list  
number_of_levels integer  
cell_list      list
```

ARGUMENTS

-from *start_point*

Specifies the starting pin or port of the fanout tree from which the buffers or inverter pairs need to be removed. Only driver pins or driver ports can be specified for this option.

This option is mutually exclusive with the **-net** and **cell_list** options.

-net *net_list*

Specifies the starting net of the fanout tree from which the buffers or inverter pairs need to be removed. Only one net can be specified for this option and the driver pin of this net would be considered as the start for fanout traversal. This means that all of the load cells connected to the net are considered in the buffer tree.

This option is mutually exclusive with the **-from** and **cell_list** options.

-to *end_point_list*

Specifies the ending pins or ports where the fanout traversal stops. Only those buffers or inverter pairs that are in the fanin path of **-to** and the fanout path of **-from** or **-net** will be removed.

This option is mutually exclusive with the **-level** option.

-level *number_of_levels*

Specifies the number of levels for buffers or inverter pairs to be removed. If you specify a *number_of_levels* value less than the number of buffers or inverters between those specified by the *start_point_list* and *end_point_list* arguments, the command removes the buffers or inverter pairs only on the number of levels from the value of *end_point_list*. Valid **-level** values are from 1 to 1,000,000. The default value is 1.

This option is mutually exclusive with the **-to** option.

cell_list

Specifies the buffer or inverter cells to be removed. Each buffer is a cell name or a collection of netlist cells. When inverter cells are specified, they must be specified in pairs.

This option is mutually exclusive with the **-to** and **-from** options.

DESCRIPTION

This command removes buffer cells when you specify a *cell_list* and the buffer tree at the specified pins, or the net on a mapped design. When a driver pin or a net is given, the buffer tree is removed with this driver as the root. The buffer-tree removal process starts at the source and removes all buffer and inverter cells in its transitive fanout until it finds nonbuffer cells or when the **-level** number of transitive fanouts is reached.

This command does not remove buffer trees across the hierarchy.

This command can remove inverters only as pairs. You must specify both of the pairing inverters if you use the *cell_list* argument. The *end_point_list* specified in the **-to** option can contain only load pins of the second inverter in the inverter pair. An inverter pair is considered as one level when decrementing the **-level** option.

This command accepts the final implementation of buffer-tree removal, even if it negatively impacts the timing or Design Rule Checking (DRC) violations in the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes a buffer cell:

```
prompt> remove_buffer -from [get_pins cell1/O]
Disconnecting net 'net1' from pin 'cell2/I'.
Disconnecting net 'net2' from pin 'cell5/I'.
Disconnecting net 'net2' from pin 'cell2/O'.
Removing net 'net2' in design 'Top'.
Connecting net 'net1' to pin 'cell5/I'.
Removing cell 'cell2' in design 'Top'.
1
```

```
prompt> remove_buffer -net [get_nets net1]
Disconnecting net 'net1' from pin 'cell2/I'.
Disconnecting net 'net2' from pin 'cell5/I'.
Disconnecting net 'net2' from pin 'cell2/O'.
```

```
Removing net 'net2' in design 'Top'.  
Connecting net 'net1' to pin 'cell5/I'.  
Removing cell 'cell2' in design 'Top'.  
1
```

```
prompt> remove_buffer cell2  
Disconnecting net 'net1' from pin 'cell2/I'.  
Disconnecting net 'net2' from pin 'cell5/I'.  
Disconnecting net 'net2' from pin 'cell2/O'.  
Removing net 'net2' in design 'Top'.  
Connecting net 'net1' to pin 'cell5/I'.  
Removing cell 'cell2' in design 'Top'.  
1
```

SEE ALSO

```
all_fanout(2)  
insert_buffer(2)  
report_buffer_tree(2)  
report_cell(2)  
report_constraint(2)
```

remove_buffer_tree

Removes the buffer tree at a specified driver pin.

SYNTAX

```
remove_buffer_tree  
  [-from pin_list]  
  [-all]  
  [-source_of patterns]
```

Data Types

pin_list list

ARGUMENTS

-from *pin_list*

Specifies a list of driver pins or nets from which buffer trees are to be removed. This option is mutually exclusive with the **-all** and **-source_of** options.

-all

Removes all buffer trees from all nets. This option is mutually exclusive with the **-from**, **-threshold** and **-source_of** options. By default, this option is off.

-source_of *patterns*

Specifies a list of load pins. The command removes the buffer trees from the global drivers of the specified pins. If the cell of a specified pin is buffer, the cell itself is also removed. This enables you to remove a buffer tree without having to identify its source.

This option is mutually exclusive with the **-all** and **-from** options.

DESCRIPTION

This command removes the buffer tree for each specified driver pin and for the driver pin of each specified net. Unlike the **create_buffer_tree** command, the specified driver pin can be a hierarchical pin.

By default, the **remove_buffer_tree** command treats hierarchical boundaries as leaf loads and does not remove buffers across the hierarchy unless you use the **-hierarchy** option.

This command works only on a placed design. Physical libraries corresponding to the given logic library must be present. The

capacitance and resistance per unit length, either derived from the physical library or specified by you, must be available.

This command does not guarantee a legal placement; however, this is the intended behavior and the optimizations generally rely on the **legalize_placement** command to legalize the design after the **remove_buffer_tree** command is issued.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes a buffer tree driven by pin a/O.

```
prompt> remove_buffer_tree -from [get_pins a/O]
```

The following example removes the buffer tree driving pin b/I.

```
prompt> remove_buffer_tree -source_of b/I
```

The following example forces all high fanout buffer trees in the design to be removed.

```
prompt> remove_buffer_tree -all
```

SEE ALSO

- `create_buffer_tree(2)`
- `report_buffer_tree(2)`
- `report_buffer_tree_qor(2)`
- `report_net_fanout(2)`
- `set_cost_priority(2)`

remove_bus

Removes a port bus or net bus.

SYNTAX

```
status remove_bus  
      object_list
```

Data Types

object_list list

ARGUMENTS

object_list

Specifies a list of port or net buses to remove.

DESCRIPTION

This command removes port or net buses from a design. If port and net buses have the same name, both port buses and net buses are removed. Individual members of buses remain. The bus can be from the current design or one of its subdesigns. To delete a bus on a subdesign, it must be unique.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the specified port buses:

```
prompt> remove_bus {AB AC}
```

The following example removes the specified net buses:

```
prompt> remove_bus {BC BD}
```

In the following example, all buses with names that end in `s` are removed. If port and net buses have the same name, both buses are removed.

```
prompt> remove_bus *s
```

In the following example, all buses with names that start with `s` in the U1/U2 instance are removed. The U1/U2 instance is a unique instantiation of the design named `bot`.

```
prompt> remove_bus U1/U2/s*
```

SEE ALSO

- `create_bus(2)`
- `create_net(2)`
- `remove_net(2)`

remove_cache

Selectively removes elements from the synthetic library cache directories.

SYNTAX

```
status remove_cache
  [-design_lib list]
  [-module list]
  [-implementation list]
  [-parameters parameter_list]
  [-tech_lib list]
  [-wire_load list]
  [-operating_conditions list]
  [-directory dir_list]
  [-smaller size | -larger size]
  [-netlist_only | -model_only]
  [-accessed_since days
   | -accessed_beyond days]
```

Data Types

<i>list</i>	<i>dir_list</i>
<i>parameter_list</i>	<i>list</i>
<i>dir_list</i>	<i>list</i>
<i>size</i>	<i>integer</i>
<i>days</i>	<i>float</i>

ARGUMENTS

-design_lib *list*

Restricts the deletion to the design libraries in the list. Strings in the list can contain the * (asterisk) wildcard character (for example, { DW_01 DW_ * *SNPS* }).

-module *list*

Restricts the deletion to the modules in the list. Strings in the list can contain the * (asterisk) wildcard character.

-implementation *list*

Restricts the deletion to the implementations in the list. Strings in the list can contain the * (asterisk) wildcard character.

-parameters *parameter_list*

Restricts the deletion to the parameters in the list. See the "DESCRIPTION" section for more information on *parameter_list*.

-tech_lib *list*

Restricts the deletion to the technology libraries in the list. Strings in the list can contain the * (asterisk) wildcard character.

-wire_load *list*

Restricts the deletion to the wire loads in the list. Strings in the list can contain the * (asterisk) wildcard character.

-operating_conditions *list*

Restricts the deletion to the operating conditions in the list. Strings in the list can contain the * (asterisk) wildcard character.

-directory *dir_list*

Uses the given list of directories, instead of the cache-read and cache-write variables, as the cache directories from which to delete.

-smaller *size*

Restricts the deletion to cache elements that are strictly smaller than *size* bytes.

-larger *size*

Restricts the deletion to cache elements that are strictly larger than *size* bytes.

-netlist_only

Deletes only netlists.

-model_only

Deletes only models.

-accessed_since *days*

Restricts the deletion to cache elements whose last access time has been from now to *days* days ago.

-accessed_beyond *days*

Restricts the deletion to cache elements whose last access time is older than *days* days.

DESCRIPTION

This command selectively removes cache elements from synthetic library cache directories. The command also removes any empty directories it encounters in the cache directories. Without command arguments, all cache elements in the cache directories are removed. If there are command arguments that restrict the deletion, the cache element is deleted only if it matches each command argument specified. An element matches a command argument if it matches one of the items in the command argument's list. Without the **-directory** command argument, the cache directories are the ones listed in the **cache_read** and **cache_write** variables.

For a description of parameter matching and an explanation of netlists and models in the cache, refer to the **report_cache** man page. Except for the various output formats in **report_cache**, any command-line option for **report_cache** can also be used for **remove_cache**.

For more information, see the DesignWare documentation set.

EXAMPLES

The following example removes the cache elements with parameters that match width=5:

```
prompt> remove_cache -parameters {"width=5"}
```

```
prompt> report_cache -dir
```

```
*****
```

```
Report : cache
```

```
Version: E-2010.12
```

```
Date : Mon Jul 19 11:10:51 2010
```

```
*****
```

```
=====
| DESIGN | MODULE | IMPLEMENT- | PARAMETERS | ACCESS | SIZE |
| LIBRARY | | ATION | | days | bytes |
=====
```

```
Cache Directory Root: '/remote/us01dwp017/dws.drg49/yingw/spf_main_ls-yingw/syn/\
unit/dw/synlib/xg_mode/tool_cache/'
```

```
Technology Library: 'lsi_10k.db'
```

```
Wire Load: no_wire_load
```

```
Operating Conditions: no_operating_conditions
```

```
| DW01 | DW01_NAND2 | str | | 0.0 | 32768 |
```

```
=====
Cache Directory Root: '/remote/us01dwp017/dws.drg49/yingw/spf_main_ls-yingw/syn/\
unit/dw/synlib/xg_mode/tool_cache/'
```

```
Technology Library: 'lsi_10k.db'
```

```
Wire Load: '-default-'
```

```
Operating Conditions: '-default-'
```

```
| DW01 | DW01_NAND2 | str | | 0.0 | 32768 |
```

```
1
```

SEE ALSO

```
create_cache(2)
report_cache(2)
report_synlib(2)
set_wire_load_min_block_size(2)
set_wire_load_mode(2)
set_wire_load_model(2)
set_wire_load_selection_group(2)
cache_read(3)
cache_write(3)
```

remove_case_analysis

Removes the case-analysis value from the specified input ports or pins.

SYNTAX

```
status remove_case_analysis  
    port_or_pin_list | -all
```

Data Types

port_or_pin_list collection

ARGUMENTS

port_or_pin_list

Specifies the ports or pins from which to remove the case analysis value. You must specify either *port_or_pin_list* or **-all**.

-all

Remove case analysis for all ports or pins in the design where the **set_case_analysis** command has previously been set. You must specify either **-all** or *port_or_pin_list*.

DESCRIPTION

The **remove_case_analysis** command removes the case-analysis value previously specified on ports or pins. Case analysis is specified using the **set_case_analysis** command. You must specify either **-all** or *port_or_pin_list*. The tool issues an error message if neither argument is specified.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example specifies that ports in0 and in2 are set to the constant logic value 0:

```
prompt> set_case_analysis 0 {in0 in2}
```

```
prompt> report_case_analysis
```

```
*****  
Report : case_analysis  
Design : middle  
Version: 1997.01-development  
Date  : Mon Jul 22 08:13:50 1996  
*****
```

Pin name	Case analysis value
in2	0
in0	0

The following example removes the case analysis entry on port in2:

```
prompt> remove_case_analysis {in2}  
prompt> report_case_analysis
```

```
*****  
Report : case_analysis  
Design : middle  
Version: 1997.01-development  
Date  : Mon Jul 22 08:14:16 1996  
*****
```

Pin name	Case analysis value
in0	0

SEE ALSO

report_case_analysis(2)
set_case_analysis(2)

remove_cell

Removes cells from the current design.

SYNTAX

```
status remove_cell  
      cell_list | -all
```

Data Types

cell_list list

ARGUMENTS

cell_list

Specifies a list of cells to be removed from the current design. Each cell name must exist in the current design. You must specify either *cell_list* or **-all**.

-all

Removes all cells in the current design. You must specify either **-all** or *cell_list*.

DESCRIPTION

The **remove_cell** command removes cells or cell instances from the current design. The command also removes pins owned by the specified cells. The design or library cell to which the cell refers is not removed. If a cell instance is used, the parent design must be unique.

To create cells, use the **create_cell** command.

EXAMPLES

The following example uses **remove_cell** to remove the specified cells in the current design:

```
prompt> get_cells ""  
{U1 U2 U3 U4 U5 U6 U7 U8}
```

```
prompt> remove_cell {U1 U2}
Removing cell 'U1' in design 'example'.
Removing cell 'U2' in design 'example'.
```

```
prompt> get_cells ""
{U3 U4 U5 U6 U7 U8}
```

The following example removes all cells remaining in the current design:

```
prompt> remove_cell -all
Removing cell 'U3' in design 'example'.
Removing cell 'U4' in design 'example'.
Removing cell 'U5' in design 'example'.
Removing cell 'U6' in design 'example'.
Removing cell 'U7' in design 'example'.
Removing cell 'U8' in design 'example'.
```

```
prompt> get_cells ""
{}
```

In the following example, the cell instances *U1/temp1* and *U1/temp2* are removed. The design named *MID* must be unique.

```
prompt> remove_cell {U1/temp1 U1/temp2}
Removing cell 'U1/temp1' in design 'MID'.
Removing cell 'U1/temp2' in design 'MID'.
```

SEE ALSO

`create_cell(2)`
`current_design(2)`

remove_cell_degradation

Removes the **cell_degradation** attribute on specified ports or designs.

SYNTAX

```
status remove_cell_degradation  
    object_list
```

Data Types

object_list list

ARGUMENTS

object_list

Specifies a list of names of input ports on which to remove the **cell_degradation** attribute.

DESCRIPTION

This command removes the **cell_degradation** attribute on the specified ports or designs.

To get information about optimization and design rule constraints, use the **report_constraint** command. To set the **cell_degradation** attribute, use the **set_cell_degradation** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the cell degradation value on the port named *input_1*:

```
prompt> remove_cell_degradation input_1
```

SEE ALSO

- all_inputs(2)
- all_outputs(2)
- characterize(2)
- compile(2)
- remove_attribute(2)
- report_constraint(2)
- set_cell_degradation(2)
- set_max_capacitance(2)
- compile_fix_cell_degradation(3)

remove_clock

Removes clocks from the current design.

SYNTAX

```
status remove_clock  
      clock_list | -all
```

Data Types

clock_list list

ARGUMENTS

clock_list

Specifies a list of clocks to remove. You must specify either *clock_list* or **-all**.

-all

Specifies to remove all clocks. You must specify either **-all** or *clock_list*.

DESCRIPTION

This command removes the specified clock objects from the current design. You must specify either *clock_list* or **-all**.

If a clock to be removed is the only member of a path group, that path group is also removed. For information about path groups, see the **group_path** command man page.

To list all clock sources in the design, use the **report_clock** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes the clock named CLK1:

```
prompt> remove_clock CLK1
```

The following example removes all clocks from the current design:

```
prompt> remove_clock -all
```

SEE ALSO

- `create_clock(2)`
- `current_design(2)`
- `group_path(2)`
- `report_clock(2)`
- `reset_design(2)`

remove_clock_gating

Directs the **compile -incremental** and **compile_ultra -incremental** commands to remove clock gating from objects clock-gated by Power Compiler.

SYNTAX

```
status remove_clock_gating
  [-gated_registers gated_register_list]
  [-min_bitwidth minsize_value]
  [-gating_cells clock_gating_cells_list]
  [-all]
  [-no_hier]
  [-verbose]
  [-undo]
```

Data Types

```
gated_register_list    list
minsize_value         integer
clock_gating_cells_list list
```

ARGUMENTS

-gated_registers *gated_register_list*

Specifies that all registers in the *gated_register_list* are ungated.

-min_bitwidth *minsize_value*

Specifies that all registers in the register banks whose size is smaller than the *minsize_value* are ungated.

-gating_cells *clock_gating_cells_list*

Specifies that all objects gated by clock-gating cells in the *clock_gating_cells_list* are ungated and removes clock-gating cells from the design.

-all

Removes clock gating from all levels of hierarchy, unless the **-no_hier** option is used. If the **-no_hier** is specified, clock gating is only removed from the top level of the current design.

-no_hier

Removes clock gating from the top level of the current design only when used with the **-all** or **-min_bitwidth** options.

-verbose

Prints additional information as the command executes.

-undo

Removes directives specified by any previously executed **remove_clock_gating** commands. Based on the specified options, this option deletes directives specified by previously-executed **remove_clock_gating** commands. Using the **-undo** option before the **compile_ultra -incremental** command modifies the netlist to remove clock gating.

DESCRIPTION

This command directs the **compile -incremental** and **compile_ultra -incremental** commands to remove clock gating from objects (registers and clock-gating cells) clock gated by Power Compiler. This command provides a mechanism to selectively remove clock gating from various parts of the design. Power Compiler performs clock gating at the register transfer level (RTL) during the execution of the **compile_ultra -gate_clock** command.

Further down the compile flow, you might have to selectively remove some clock-gating elements from the design. The **remove_clock_gating** command provides a mechanism to do so without having to start at the RTL again.

This command provides directives to the **compile** and **compile_ultra** commands used with the **-incremental** option that perform circuit modifications to remove clock gating from the design. See the man pages for the **compile** and **compile_ultra** commands for more information on the **-incremental** options. This command also removes redundant clock-gating cells that do not gate any clock-gating cells. Any associated test-observation logic is removed during optimization. All registers that are ungated are remapped to new sequential cells. This might result in new pin names for the registers. A register can be clock-gated by multiple clock gates in a multistage clock-gating design. If the register is ungated, all clock gating on the clock path for this register is removed. However, if a single clock gate is specified for removal, the remaining clock-gating cells still exist on the clock path of the register.

Pin-based timing exceptions set on the pins of the old register might have been set by the **set_max_delay**, **set_min_delay**, **set_multicycle_path** and **set_false_path** commands. These exceptions might not be properly transferred during the transformation if the new and old pin names do not match. The **remove_clock_gating** command issues a warning if there are pin-based timing exceptions on the register to be ungated.

It is good practice to run the **report_timing** command again to confirm that the pin-based exceptions are being properly applied after the registers are ungated during execution of the **compile** command with the **-incremental** option.

This command has no effect on a clock-gating cell or its gated registers if the clock gate or its parent hierarchical cell is marked **dont_touch** with the **set_dont_touch** command.

EXAMPLES

The following example provides a directive to the **compile_ultra -incremental** command to remove clock gating from all registers gated by the cell *clk_gate_pipeline_reg_A* and removes the cell:

```
prompt> remove_clock_gating -gating_cell clk_gate_pipeline_reg_A
```

The following example provides a directive to the **compile_ultra -incremental** command to remove clock gating from the gated registers *pipeline_reg_A[0]* and *pipeline_reg_A[2]*. If the clock-gating cell that gates these registers does not drive any other gating cell, it is also removed.

```
prompt> remove_clock_gating \
      -gated_registers {pipeline_reg_A[0] pipeline_reg_A[2]}
```

The following example undoes the effect of the directive in the first example above, to remove the clock-gating cell

clk_gate_pipeline_reg_A and ungate all registers gated by this cell. Run this command before running the **compile_ultra -incremental** command.

```
prompt> remove_clock_gating -undo \  
-gating_cell clk_gate_pipeline_reg_A
```

The following example provides a directive to the **compile_ultra -incremental** command to remove clock gating from all gated registers and to remove all clock-gating cells from the current level of the design hierarchy:

```
prompt> remove_clock_gating -all -no_hier
```

The following example provides a directive to the **compile_ultra -incremental** command to remove clock gating from all gated registers and remove all clock-gating cells from all designs at or below the current level of the design hierarchy:

```
prompt> remove_clock_gating -all
```

SEE ALSO

- compile(2)
- compile_ultra(2)
- report_clock_gating(2)
- set_clock_gating_style(2)
- set_false_path(2)
- set_max_delay(2)
- set_min_delay(2)
- set_multicycle_path(2)

remove_clock_gating_check

Removes setup and hold checks from the specified clock-gating cells.

SYNTAX

```
status remove_clock_gating_check  
  [-setup]  
  [-hold]  
  [-rise]  
  [-fall]  
  object_list
```

Data Types

object_list list

ARGUMENTS

-setup

Removes setup checks from the specified clock-gating cells.

-hold

Removes hold checks from the specified clock-gating cells.

-rise

Removes the clock-gating constraint on the rising delays. If you do not specify either **-rise** or **-fall**, the default is to remove constraints on both rising and falling delays.

-fall

Removes the clock-gating constraint on the falling delays. If you do not specify either **-rise** or **-fall**, the default is to remove constraints on both rising and falling delays.

If you omit the **-setup** and **-hold** options, by default both types of checks are removed.

object_list

Specifies a list of designs, cells, pins, or clock objects from which to remove the clock-gating checks.

DESCRIPTION

The **remove_clock_gating_check** command removes clock-gating checks that were added using the **set_clock_gating_check** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes setup and hold checks from the CHIP1 design:

```
prompt> remove_clock_gating_check CHIP1
```

The following example removes only the setup checks on the clock-gating inputs of the CLK_GATE1 cell in the current design:

```
prompt> remove_clock_gating_check -setup CLK_GATE1
```

The following example removes only the fall hold checks on the clock-gating inputs of the CLK_GATE2 cell in the current design:

```
prompt> remove_clock_gating_check -fall -hold CLK_GATE2
```

SEE ALSO

- create_clock(2)
- current_design(2)
- report_clock(2)
- set_clock_gating_check(2)

remove_clock_gating_style

Removes the clock gating style applied on a given set of hierarchical cells or power domains.

SYNTAX

```
status remove_clock_gating_style  
  [-instances {cell_list}]  
  [-power_domains {power_domain_list}]  
  [-designs designs]
```

Data Types

```
cell_list      list  
power_domain_list list
```

ARGUMENTS

-instances {*cell_list*}

Specifies that the clock gating style should be removed from the given list of hierarchical cells.

-power_domains {*power_domain_list*}

Specifies that the clock gating style should be removed from the given list of power domains.

DESCRIPTION

This command removes the clock gating style previously specified with command **set_clock_gating_style** (with instance specific clock gating style enabled) on a given hierarchical cell, power domain or design. When the clock gating style is removed from a certain hierarchical instance, the style used for that instance is obtained according to the precedence described for **set_clock_gating_style** on section **Instance specific Clock-gating Style**.

When using this command with none of the options **-instances** and **-power_domains**, the style removed is the one previously set on the **current_design**.

The use of this command is subject to have instance specific clock gating style enabled, by setting variable **power_cg_iscgs_enable** to **true**.

EXAMPLES

The following example removes the clock gating style from instances **m1/b1** and **m2**.

```
prompt> remove_clock_gating_style -instances {m1/b1 m2}
```

SEE ALSO

- compile(2)
- compile_ultra(2)
- set_clock_gating_style(2)

remove_clock_groups

Removes specific exclusive or asynchronous clock groups from the current design.

SYNTAX

```
status remove_clock_groups  
-logically_exclusive  
| -asynchronous  
| -physically_exclusive  
name_list | -all
```

Data Types

name_list list

ARGUMENTS

-logically_exclusive

Specifies that groups set for logically-exclusive clocks are to be removed. You must specify either **-logically_exclusive**, **-asynchronous**, or **-physically_exclusive**.

-asynchronous

Specifies that groups set for asynchronous clocks are to be removed. You must specify either **-asynchronous**, **-logically_exclusive**, or **-physically_exclusive**.

-physically_exclusive

Specifies that groups set for physically-exclusive clocks are to be removed. You must specify either **-physically_exclusive**, **-logically_exclusive**, or **-asynchronous**.

name_list

Specifies a list of clock groups to be removed, which matches the groups in the given names. Use the **set_clock_groups** command to predefine these names. Substitute the list you want for *name_list*. You must specify either the *name_list* argument or the **-all** option.

-all

Specifies to remove all groups set for exclusive or asynchronous clocks in the current design. You must specify either the **-all** option or the *name_list* argument.

DESCRIPTION

This command removes specific exclusive or asynchronous clock groups from the current design. These clock groups are specified by the *name_list* from the current design. You must specify either **-logically_exclusive**, **-physically_exclusive**, or **-asynchronous**. You must specify either *name_list* or **-all**.

To find the names for existing groups, use the **report_clock** command with the **-groups** option.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes logically-exclusive clock groups named mux:

```
prompt> remove_clock_groups -logically_exclusive mux
```

The following example removes all asynchronous clock groups from the current design:

```
prompt> remove_clock_groups -asynchronous -all
```

SEE ALSO

report_clock(2)
set_clock_groups(2)

remove_clock_latency

Removes clock latency information from the specified objects.

SYNTAX

```
string remove_clock_latency  
  [-fall]  
  [-min]  
  [-max]  
  [-source]  
  [-early]  
  [-late]  
  object_list  
  [-rise]
```

Data Types

object_list list

ARGUMENTS

-fall

Specifies that the fall clock latency should be removed. By default, this is removed for both minimum and maximum clock latency.

-min

Specifies that the minimum clock latency should be removed. By default, this is removed for both rise and fall clock latency.

-max

Specifies that the maximum clock latency should be removed. By default, this is removed for both rise and fall clock latency.

-source

Specifies that clock source latency should be removed.

-early

Specifies that the late-clock source latency should be removed. If this option is not specified, both early and late clock latencies are removed.

-late

Specifies that the late clock source latency should be removed. If this option is not specified, both early and late clock latencies are removed.

object_list

Provides a list of clocks, ports, or pins.

-rise

Specifies that the rise clock latency should be removed. By default, this is removed for both minimum and maximum clock latency.

DESCRIPTION

The **remove_clock_latency** command removes user-specified clock network or clock-source latency information from specified objects. Clock-network latency is the time it takes for a clock signal on the design to propagate from the clock definition point to a register clock pin. Clock-source latency (also called insertion delay) is the time it takes for a clock signal to propagate from its actual ideal waveform origin point to the clock definition point in the design. Clock-network and clock-source latency information is set on objects using the **set_clock_latency** command. See the **set_clock_latency** man page for more information.

You can remove selected portions of the clock latency by specifying options such as **-rise**, **-fall**, **-min**, **-max**, **-late**, and **-early**.

To report clock-network and clock-source latency information, use the **report_clock** command with the **-skew** option.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes clock latency information from the clock named CLK1:

```
prompt> remove_clock_latency [get_clocks CLK1]
```

The following example removes clock-source latency information from the CLK1 clock:

```
prompt> remove_clock_latency -source [get_clocks CLK1]
```

The following example removes only rise clock-source latency information from the CLK1 clock:

```
prompt> remove_clock_latency -rise -source [get_clocks CLK1]
```

SEE ALSO

create_clock(2)
remove_attribute(2)
remove_clock(2)
remove_clock_uncertainty(2)
report_clock(2)
reset_design(2)
set_clock_latency(2)
set_clock_uncertainty(2)

```
set_input_delay(2)
```

remove_clock_sense

Removes clock sense information from the specified pins.

SYNTAX

```
status remove_clock_sense  
  [-all]  
  [-clocks clock_list]  
  pins
```

Data Types

```
clock_list  list  
pins        list
```

ARGUMENTS

-all

Removes all clock sense information in the current design.

-clocks *clock_list*

Specifies the list of clocks for which the clock-sense information is removed. This option can remove only clock-sense information that has been set by using the **set_clock_sense** command with the **-clocks** option. It does not remove clock-sense settings from pins. By default, the clock-sense information is removed for all clocks passing through the specified pins.

pins

Specifies the pins from which to remove clock-sense information.

DESCRIPTION

This command removes user-specified clock-sense (unateness) information from specified pins and clocks. To set clock-sense information, use the **set_clock_sense** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes positive unateness defined on a pin named XOR/Z with respect to clock CLK1, but does not remove the negative unateness:

```
prompt> set_clock_sense -positive -clocks [get_clock CLK1] XOR/Z  
prompt> set_clock_sense -negative XOR/Z  
prompt> remove_clock_sense -clocks [get_clocks CLK1] XOR/Z
```

The following example removes all unateness information in the current design;

```
prompt> remove_clock_sense -all
```

SEE ALSO

`set_clock_sense(2)`

remove_clock_transition

Removes clock-transition attributes on the specified clock objects.

SYNTAX

```
status remove_clock_transition  
      clock_list
```

ARGUMENTS

clock_list

Specifies the clocks from which to remove the transition attributes.

DESCRIPTION

This command removes the clock-transition attributes on the specified clocks.

To list all clock-transition values that have been set, use the **report_clock -skew** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes transition attributes on the clock named CLK:

```
prompt> remove_clock_transition CLK
```

The following example removes transition attributes on all clocks in the current design:

```
prompt> remove_clock_transition [all_clocks]
```

SEE ALSO

`create_clock(2)`
`current_design(2)`
`report_clock(2)`
`reset_design(2)`
`set_clock_transition(2)`

remove_clock_uncertainty

Removes clock-uncertainty information previously set by the **set_clock_uncertainty** command.

SYNTAX

```
string remove_clock_uncertainty
  [object_list
  | -from from_clock
  | -rise_from rise_from_clock
  | -fall_from fall_from_clock
  -to to_clock
  | -rise_to rise_to_clock
  | -fall_to fall_to_clock]
  [-rise]
  [-fall]
  [-setup]
  [-hold]
```

Data Types

```
object_list    list
from_clock     list
rise_from_clock list
fall_from_clock list
to_clock       list
rise_to_clock  list
fall_to_clock  list
```

ARGUMENTS

object_list

Specifies a list of clocks, ports, or pins from which uncertainty information is to be removed. By default, uncertainty information is removed from all objects in the current design. Use either the pair of **-from/-rise_from/-fall_from** and **-to/-rise_to/-fall_to** options or the *object_list* option. These options are mutually exclusive; specify only one.

-from *from_clock*

Specifies the source and destination clocks for interclock uncertainty. Specify either the pair of **-from/-rise_from/-fall_from** and **-to/-rise_to/-fall_to**, or *object_list*. These options are mutually exclusive; specify only one.

-rise_from *rise_from_clock*

Specifies the source and destination clocks, but indicates that *uncertainty* applies only to the rising edge of the source clock. Use only one of the **-from**, **-rise_from**, or **-fall_from** options.

-fall_from fall_from_clock

Specifies the source and destination clocks, but indicates that *uncertainty* applies only to the falling edge of the source clock. Use only one of the **-from**, **-rise_from**, or **-fall_from** options.

-to to_clock

Specifies the destination clocks for interclock uncertainty. Specify either the pair of **-from/-rise_from/-fall_from** and **-to/-rise_to/-fall_to**, or *object_list*. These options are mutually exclusive; specify only one.

-rise_to rise_to_clock

Specifies the destination clocks for interclock uncertainty, but indicates that *uncertainty* applies only to the rising edge of the destination clock. Use only one of the **-to**, **-rise_to**, or **-fall_to** options.

-fall_to fall_to_clock

Specifies the destination clocks for interclock uncertainty, but indicates that *uncertainty* applies only to the falling edge of the destination clock. Use only one of the **-to**, **-rise_to**, or **-fall_to** options.

-rise

Specifies that uncertainty is to be removed for only the rising clock edge. By default, uncertainty is removed for both rising and falling clock edges. This option is valid only for interclock uncertainty, and is now obsolete. Unless you need this option for backward-compatibility, use **-rise_to** instead.

-fall

Specifies that uncertainty is to be removed for only the falling clock edge. By default, uncertainty is removed for both rising and falling clock edges. This option is valid only for interclock uncertainty, and is now obsolete. Unless you need this option for backward-compatibility, use **-fall_to** instead.

-setup

Specifies that only setup check uncertainty is to be removed. By default, both setup and hold check uncertainties are removed.

-hold

Specifies that only hold check uncertainty is to be removed. By default, both setup and hold check uncertainties are removed.

DESCRIPTION

This command removes clock uncertainty information previously set by the **set_clock_uncertainty** command from clocks, ports, or pins, or between specified clocks. If the command is issued without options, all clock uncertainty information is removed from the current design. To display clock uncertainty information, use the **report_clock** command with the **-skew** option.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes uncertainty information from a clock named CLK and a pin named clk_buf/Z:

```
remove_clock_uncertainty
```

```
prompt> remove_clock_uncertainty [get_clocks CLK]  
prompt> remove_clock_uncertainty [get_pins clk_buf/Z]
```

The following example removes interclock uncertainties between the PHI1 and PHI2 clock domains:

```
prompt> remove_clock_uncertainty -from PHI1 -to PHI1  
prompt> remove_clock_uncertainty -from PHI2 -to PHI2  
prompt> remove_clock_uncertainty -from PHI1 -to PHI2  
prompt> remove_clock_uncertainty -from PHI2 -to PHI1
```

SEE ALSO

- `all_clocks(2)`
- `create_clock(2)`
- `report_clock(2)`
- `set_clock_latency(2)`
- `set_clock_transition(2)`
- `set_clock_uncertainty(2)`

remove_congestion_options

Removes congestion options from the current design.

SYNTAX

```
status remove_congestion_options  
[-all]  
id_list
```

Data Types

id_list list

ARGUMENTS

-all

Removes all congestion options from the current design.

id_list

Removes congestion options with the specified IDs.

DESCRIPTION

The **remove_congestion_options** command removes the congestion options from the current design. If the **-all** option is specified, all congestion options are removed from the design. If a list of IDs is provided, the congestion options with the specified IDs are removed. The ID is displayed every time a regional congestion option is created, or it can be viewed by using the **report_congestion_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all congestion options:

```
prompt> remove_congestion_options -all
```

SEE ALSO

report_congestion(2)
report_congestion_options(2)
set_congestion_options(2)

remove_constraint

Removes all constraint attributes, clocks, and path delay information from the current design. Note that this command will be obsolete in the next release. It will be replaced by the **remove_sdc** command. Please adjust your scripts accordingly.

SYNTAX

```
status remove_constraint
  [-all]
  [-all_sdc]
```

ARGUMENTS

-all

Removes all target values from the current design.

-all_sdc

Removes all SDC constraints.

DESCRIPTION

This command removes **max_area**, **max_power**, **max_fanout**, and **max_transition** attributes from the current design. It also removes clock objects and all path delay information created by the **set_max_delay**, **set_min_delay**, **set_false_path**, **set_max_time_borrow**, and **set_multicycle_path** commands.

You can undo individual commands by using the **remove_attribute** or **reset_path** command. To remove clocks, use the **remove_clock** command.

The **reset_design** command removes all constraints and all attributes.

Multicorner-Multimode Support

This command applies to all active scenarios.

EXAMPLES

The following example removes all constraint information from the current design:


```
prompt> remove_constraint -all
```

SEE ALSO

- current_design(2)
- remove_attribute(2)
- remove_clock(2)
- report_constraint(2)
- reset_design(2)
- reset_path(2)
- set_max_area(2)
- set_max_delay(2)
- set_max_fanout(2)
- set_max_time_borrow(2)
- set_max_transition(2)

remove_core_area

Removes the core area of the design.

SYNTAX

status **remove_core_area**

ARGUMENTS

The **remove_core_area** command has no arguments.

DESCRIPTION

This command removes the core area of the design.

EXAMPLES

The following example removes the core area of the current design:

```
prompt> remove_core_area  
1
```

SEE ALSO

`create_core_area(2)`

remove_data_check

Removes specified data-to-data checks previously set by the **set_data_check** command.

SYNTAX

```
string remove_data_check  
  -from from_object  
    | -rise_from from_object  
    | -fall_from from_object  
  -to to_object  
    | -rise_to to_object  
    | -fall_to to_object  
  [-setup | -hold]  
  [-clock clock]
```

Data Types

from_object collection
to_object collection

ARGUMENTS

-from *from_object*

Specifies a pin or port in the current design as the related pin of the data-to-data check to be removed. Both rising and falling checks are removed. You must specify one of **-from**, **-rise_from**, or **-fall_from**.

-rise_from *from_object*

Specifies a pin or port in the current design as the related pin of the data-to-data check to be removed. This is similar to the **-from** option, but applies to only rising delays at the related pin. You must specify one of **-from**, **-rise_from**, or **-fall_from**.

-fall_from *from_object*

Specifies a pin or port in the current design as the related pin of the data-to-data check to be removed. This is similar to the **-from** option, but applies to only falling delays at the related pin. You must specify one of **-from**, **-rise_from**, or **-fall_from**.

-to *to_object*

Specifies a pin or port in the current design as the constrained pin of the data-to-data check to be created. Both rising and falling checks are removed. You must specify one of **-to**, **-rise_to**, or **-fall_to**.

-rise_to *to_object*

Specifies a pin or port in the current design as the constrained pin of the data-to-data check to be created. This is similar to the **-to** option, but applies to only rising delays at the constrained pin. You must specify one of **-to**, **-rise_to**, or **-fall_to**.

-fall_to to_object

Specifies a pin or port in the current design as the constrained pin of the data-to-data check to be created. This is similar to the **-to** option, but applies to only falling delays at the constrained pin. You must specify one of **-to**, **-rise_to**, or **-fall_to**.

-setup

Indicates that only the setup data check is to be removed. If neither **-setup** nor **-hold** is specified, both setup and hold checks are removed.

-hold

Indicates that only the hold data check is to be removed. If neither **-setup** nor **-hold** is specified, both setup and hold checks are removed.

-clock clock

Indicates that the data check for the specified clock at the related pin is to be removed. This option applies only if a previous **set_data_check** command used the **-clock** option to specify the same clock; otherwise, this option is ignored.

DESCRIPTION

The **remove_data_check** command specifies data-to-data check(s) to be removed between the *from* object and the *to* object for the specified options. These checks were previously set using the **set_data_check** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes a data-to-data check from and1/B to and1/A with respect to the rising edge of signals at and1/B. Both setup and hold checks are removed.

```
prompt> remove_data_check -rise_from and1/B -to and1/A
```

The following example removes setup data-to-data checks between and1/B and and1/A with respect to the rising edge of signals at and1/B and the falling edge of signals at and1/A, coming from startpoints triggered by the CK1 clock:

```
prompt> remove_data_check -rise_from and1/B \
-fall_to and1/A -setup -clock [get_clock CK1]
```

SEE ALSO

report_timing(2)
 set_data_check(2)
 update_timing(2)
 timing_enable_multiple_clocks_per_reg(3)

remove_design

Removes designs or libraries from memory.

SYNTAX

```
status remove_design
      [design_list | -designs | -all]
      [-hierarchy]
      [-quiet]
```

Data Types

design_list list

ARGUMENTS

design_list

Specifies the designs or libraries to remove. The default is the current design. This argument cannot be used with the **-designs** or **-all** options. You can specify a design to be removed with or without an absolute path. To determine the absolute path of designs, use the **list_files** command.

-designs

Removes all designs in memory. This option cannot be used with the *design_list* argument or the **-all** option.

-all

Removes all designs and libraries in memory. This option cannot be used with the *design_list* argument or the **-designs** option.

-hierarchy

Removes all designs within the hierarchy of the designs in the *design_list* argument or in the current design. The default is not to remove any subdesigns.

-quiet

Turns off verbose mode that prints out messages showing the names of the designs or libraries that are being removed.

DESCRIPTION

This command removes designs or libraries from memory. If you do not specify any arguments, the current design is removed. When

a design or library is removed, the memory it occupies is freed.

If all designs are removed from memory, then any scenarios that exist are automatically deleted. If necessary, **create_scenario** can be used to re-create deleted scenarios.

Note that memory is freed to be reused by this same process. However, memory is not returned to the operating system until you exit the tool.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information. However, if all designs are removed, all scenarios are also removed.

EXAMPLES

The following example removes the current design:

```
prompt> current_design TEST  
Current design is now 'TEST'
```

```
prompt> remove_design  
Removing design 'TEST'
```

The following example removes a specified design and all subdesigns in the hierarchy:

```
prompt> remove_design -hierarchy TEST2  
Removing design 'TEST2'  
Removing design 'ADDER'  
Removing design 'MUX8'  
Removing design 'ND17'
```

The following example removes the design named RIGHT in the file named TOP.ddc:

```
prompt> remove_design {"/remote/user/designs/TOP.ddc:RIGHT"}  
Removing design 'RIGHT'
```

The following example removes all design and library objects from memory:

```
prompt> remove_design -all  
Removing design 'A'  
Removing design 'B'  
Removing library 'test_lib'
```

The following example shows that scenarios are deleted as a side-effect of deleting all designs from memory:

```
prompt> remove_design -designs  
Removing design 'top'  
Removing design 'mid'  
Removing design 'bot'  
Information: Removing all scenarios because all designs have been  
removed. (UID-1040)  
Information: Removed Scenario S1. (UID-1024)  
Information: Removed Scenario S2. (UID-1024)  
Information: Removed Scenario S3. (UID-1024)
```

SEE ALSO

create_scenario(2)
current_design(2)
rename_design(2)
list_files(2)
UID-1040(n)

remove_dft_clock_gating_pin

Removes all DFT clock-gating pin specifications for the current design.

SYNTAX

status **remove_dft_clock_gating_pin**

ARGUMENTS

The **remove_dft_clock_gating_pin** command takes no arguments.

DESCRIPTION

The **remove_dft_clock_gating_pin** command removes all DFT clock-gating pin specifications that were defined by the **set_dft_clock_gating_pin** command for the current design.

EXAMPLES

In the following example, the command removes all the DFT clock-gating pin specifications for the current design:

```
prompt> remove_dft_clock_gating_pin
```

SEE ALSO

set_dft_clock_gating_pin(2)
report_dft_clock_gating_pin(2)

remove_dft_connect

Removes existing DFT connectivity specifications.

SYNTAX

```
status remove_dft_connect
      [-all]
      [label_name]
```

Data Types

label_name string

ARGUMENTS

-all

Removes all connectivity specifications.

label_name

Removes only the connectivity specification with the associated *label_name*.

DESCRIPTION

The **remove_dft_connect** command is used to remove DFT connectivity associations specified by the **set_dft_connect** command.

To remove a specific connectivity specification, specify the *label_name* of the association as the argument.

To remove all connectivity specifications, specify **-all**.

EXAMPLES

The following example first sets 3 connectivity specifications, then removes the specification associated with label *CON2*, and finally removes all specified connectivity by using the **-all** option:

```
prompt> set_dft_connect CON1 -source SE1 -type clock_gating_control \
      -target [list U1 U2]
```

```
prompt> set_dft_connect CON2 -source SE2 -type clock_gating_control \  
-target [list U3 U4] -exclude U3/u
```

```
prompt> set_dft_connect CON3 -source SE3 -type scan_enable \  
-target [list ff1 ff2]
```

```
prompt> remove_dft_connect CON2
```

```
prompt> remove_dft_connect -all
```

SEE ALSO

- report_dft_connect(2)
- set_dft_connect(2)
- set_dft_signal(2)

remove_dft_design

Removes the specified design so that it cannot be used for DFT insertion.

SYNTAX

```
status remove_dft_design  
-design_name design_name  
| -type design_type  
| -all
```

Data Types

```
design_name  string  
design_type  string
```

ARGUMENTS

-design_name *design_name*

Identifies the design to be removed. This option is mutually exclusive with **-type** and **-all**.

-type *design_type*

Specifies the type of the DFT design to be removed. Valid types are DECODE, CONTROL_FORCE, Z_CONTROL_FORCE, OBSERV_DGEN, SHIFT_REG, BC_1, BC_2, BC_3, BC_4, BC_5, BC_7, BC_8, BC_9, BC_10, WC_D1, WC_D1_S, WC_S1, WC_S1_S, INSTRREG, TAP, TAP_UC, LBIST_CONTROLLER, LBIST_CODEEC, LBIST_XCONTROLLER, LBIST_XCODEC, CLK_MUX, CLK_CHAIN, MBIST_CONTROLLER, MBIST_WRAPPER, and PAD.

This option is mutually exclusive with **-design_name** and **-all**.

-all

Removes all DFT designs. This option is mutually exclusive with **-design_name** and **-type**.

DESCRIPTION

The **remove_dft_design** command removes a design so that it is not used for DFT insertion.

You must use one of the three options: **-design_name**, **-type**, or **-all**. These options are mutually exclusive; use only one option for each invocation of the command.

EXAMPLES

The following example removes *my_des* so that it is not used for DFT insertion:

```
prompt> define_dft_design -design_name my_des \  
-type WC_D1 -interface \  
{shift_clk capture_clk h capture_en capture_en h \  
shift_en shift_dr h cti si h cto so h cfi \  
data_in h cfo data_out h
```

```
prompt> ....
```

```
prompt> remove_dft_design -design_name my_des
```

The following example removes all DFT designs of type *BC_4*:

```
prompt> define_dft_design -design my_des1 -type BC_4 \  
-interface {capture_clk cap_clk h capture_en cen h \  
shift_dr shift h si ti h so to h data_in di h data_out do h
```

```
prompt> define_dft_design -design my_des2 -type BC_4 \  
-interface {capture_clk cap_clk h capture_en cen h \  
shift_dr shift h si ti h so to h data_in di h data_out do h
```

```
prompt> remove_dft_design -type BC_4
```

SEE ALSO

define_dft_design(2)
insert_dft(2)
preview_dft(2)
report_dft_design(2)

remove_dft_equivalent_signals

Removes all equivalent DFT signals specified with the **set_dft_equivalent_signals** command for the specified primary signal commands.

SYNTAX

```
status remove_dft_equivalent_signals  
      primary_signal
```

ARGUMENTS

primary_signal

Specifies the primary signal used in **set_dft_equivalent_signals** to specify valid equivalences for the set of signals.

DESCRIPTION

This command removes the equivalent set of signals specified using the **set_dft_equivalent_signals** command. Use this command to remove a given equivalence specified through **set_dft_equivalent_signals**.

EXAMPLES

The following example illustrates removing equivalent scan signals for *clk1*:

```
prompt> remove_dft_equivalent_signals clk1
```

SEE ALSO

```
insert_dft(2)  
preview_dft(2)  
report_dft_equivalent_signals(2)  
set_dft_equivalent_signals(2)  
set_dft_signal(2)
```

remove_dft_location

Removes DFT hierarchy location specifications for the current design.

SYNTAX

```
status remove_dft_location
      [-type logic_type]
```

Data Types

logic_type string

ARGUMENTS

-type *logic_type*

Specifies the types of DFT hierarchy location specifications to be removed from the current design. The following values are valid for *logic_type*:

```
CODEC
TCM
LOCKUP_LATCH
PIPELINE_SI_LOGIC
PIPELINE_SE_LOGIC
WRAPPER
BSR
TAP
REC_MUX
PLL
SERIAL_CNTRL
SERIAL_REG
XOR_SELECT
RETIMING_FLOP
IEEE_1500
```

You can specify a list of multiple types with this option.

To remove a default DFT hierarchy location specification that was applied using the **set_dft_location** command without the **-include** option, do not specify a value for this option.

DESCRIPTION

The **remove_dft_location** command removes DFT hierarchy location specifications for the current design.

EXAMPLES

The following example removes the DFT hierarchy location for the design named *top*:

```
prompt> current_design top
```

```
prompt> set_dft_location -include {CODEC} core_inst/CODEC
```

```
prompt> remove_dft_location -type {CODEC}
```

```
Removed DFT location 'core_inst/CODEC' for type 'CODEC' for design 'top'.
```

```
1
```

SEE ALSO

[insert_dft\(2\)](#)

[report_dft_location\(2\)](#)

[set_dft_location\(2\)](#)

remove_dft_partition

Permanently removes a list of design DFT partitions associated with a design.

SYNTAX

```
status remove_dft_partition  
      [list_of_partition_labels]
```

Data Types

list_of_partition_labels list

ARGUMENTS

list_of_partition_labels

Specifies the partitions to be removed from the specification.

DESCRIPTION

This command permanently removes a list of design DFT partitions associated with a design.

EXAMPLES

The following example removes the partitions defined as *part1* and *part2*:

```
prompt> define_dft_partition part1 -instances {U1 U2}
```

```
prompt> define_dft_partition part2 -instances {U3 U4}
```

```
prompt> remove_dft_partition {part1 part2}
```

SEE ALSO

```
current_dft_partition(2)
define_dft_partition(2)
insert_dft(2)
preview_dft(2)
report_dft_partition(2)
set_dft_signal(2)
set_scan_compression_configuration(2)
set_scan_configuration(2)
set_scan_path(2)
```

remove_dft_power_control

Removes the power controller block specification from the current design.

SYNTAX

status **remove_dft_power_control**

ARGUMENTS

The **remove_dft_power_control** command has no arguments.

DESCRIPTION

This command removes the power controller block specification from the current design.

Use the **set_dft_power_control** command to specify a power controller block specification. Use the **report_dft_power_control** command to report the current power controller block specification.

For more information, see the man page for the **set_dft_power_control** command.

EXAMPLES

In the following example, the current power controller block specification is removed:

```
prompt> remove_dft_power_control
```

SEE ALSO

report_dft_power_control(2)
set_dft_power_control(2)

remove_dft_signal

Removes from specified ports the attributes that identify those ports as DFT signals in the current design.

SYNTAX

```
status remove_dft_signal  
-view existing_dft | spec  
-port port_list  
-test_mode mode_name_list  
[-hookup_pin pin_name]  
[-usage use_type]
```

Data Types

```
port_list      list  
mode_name_list list  
pin_name      string
```

ARGUMENTS

-view existing_dft | spec

Indicates the view to which the command applies. Valid views are **existing_dft** and **spec**.

Setting the view to **existing_dft** indicates that the command refers to the existing usage of a port. Use **existing_dft** to remove a specification made with the **set_dft_signal -view existing_dft** command.

Setting the view to **spec**, the default, indicates that the command refers to ports that the tool must use during DFT insertion. Use **spec** to remove a specification made with the **set_dft_signal -view spec** command.

This argument is required.

-port port_list

Specifies a list of ports to which the command applies.

-test_mode mode_name_list

Specifies the mode(s) to which the command applies. The default mode is **Internal_scan**. Specifying **all** as the *mode_list* indicates that the specification applies to all modes. Any mode other than the default mode or **all** must be created before running the command with this option.

-hookup_pin pin_name

Specifies the hookup pin to which the command applies.

DESCRIPTION

The **remove_dft_signal** command removes from specified ports the attributes that identify those ports as DFT signal ports in the current design. These attributes were placed on the ports with the **set_dft_signal** command.

EXAMPLES

The following example removes the DFT signal attributes from the *TM* port:

```
prompt> remove_dft_signal -port TM
```

SEE ALSO

insert_dft(2)
preview_dft(2)
report_dft_signal(2)
set_dft_signal(2)

remove_die_area

Removes the die area from the current design.

SYNTAX

```
status remove_die_area  
[-verbose]
```

ARGUMENTS

-verbose

Prints additional debugging messages.

DESCRIPTION

This command removes the die area from the current design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the die area from current design:

```
prompt> remove_die_area  
1
```

SEE ALSO

```
create_die_area(2)  
get_die_area(2)
```

remove_disable_clock_gating_check

Restores clock-gating checks that were previously disabled by the **set_disable_clock_gating_check** command for the specified cells and pins.

SYNTAX

```
status remove_disable_clock_gating_check  
      object_list
```

Data Types

object_list collection

ARGUMENTS

object_list

Specifies the cells and pins for which previously-disabled clock-gating checks are to be restored.

DESCRIPTION

This command restores clock-gating checks that were previously disabled by the **set_disable_clock_gating_check** command for the specified cells and pins.

Multicorner-Multimode Support

This command applies to all scenarios.

EXAMPLES

The following example restores disabled clock-gating checks for the an1/A pin:

```
prompt> remove_disable_clock_gating_check an1/A
```

The following example restores all disabled gating checks on the cells named U1 or U2:

```
prompt> remove_disable_clock_gating_check [get_cells {U1 U2}]
```

SEE ALSO

`set_disable_clock_gating_check(2)`

remove_disable_timing

Enables previously user-disabled timing arcs in the current design. It is equivalent to the **set_disable_timing -restore** command.

SYNTAX

```
status remove_disable_timing  
  object_list  
  [-from from_pin_name -to to_pin_name]  
  [-all_loop_breaking]
```

Data Types

```
object_list    list  
from_pin_name  string  
to_pin_name    string
```

ARGUMENTS

object_list

Specifies a list of cells, pins, ports, library pins, or library cells that are to be enabled. The cells or the cells of pins are no longer size only if you did not set size only.

-from *from_pin_name*

Specifies that arcs only from this pin on the specified cell are to be enabled. The *object_list* must contain only cells if **-from** and **-to** are specified.

-to *to_pin_name*

Specifies that arcs only to this pin on the specified cell are to be enabled. The *object_list* must contain only cells if **-from** and **-to** are specified.

-all_loop_breaking

Re-enables all stored loop-breaking timing arcs. This option applies to the entire design.

DESCRIPTION

The **remove_disable_timing** command enables timing through the specified cells, pins, or ports in the current design.

Any cell, pin, or port in the current design or its subdesigns can be disabled and thus re-enabled.

Cells at lower levels in the design hierarchy are specified as *instance1/instance2/cell_name*.

Pins at lower levels in the design hierarchy are specified as *instance1/instance2/cell_name/pin_name*.

Ports at lower levels in the design hierarchy are specified as *instance1/instance2/cell_name/port_name*.

Library pins and library cells can also be disabled and re-enabled.

Note that for subdesigns in the hierarchy, you must specify instance names instead of design names.

In disabling, when the timing through a cell is disabled, only those cell arcs that lead to the output pins of the cell are disabled. When the timing through a pin or port is disabled, only those cell arcs that lead to or from that pin or port are disabled. The **remove_disable_timing** command has a reverse effect.

When the **-from** and **-to** pins are specified, all arcs between these pins on the cell are enabled.

Use **report_lib_timing_arcs** to list the timing arcs for a library cell. Use **report_design** to list the disabled arcs in the current design so as to check the enabling effect of the **remove_disable_timing** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

In the following example, all timing arcs between pins A and Z on cell U1/U1 in the current design are enabled:

```
prompt> remove_disable_timing U1/U1 -from A -to Z
```

In the following example, all timing arcs between pins D and Q on cell FD1 in the library named my_lib are enabled. This command enables arcs on a library cell; therefore, all instances of that library cell are affected in any design linked to the same library.

```
prompt> remove_disable_timing my_lib/FD1 -from D -to Q
```

SEE ALSO

current_design(2)
remove_attribute(2)
report_lib(2)
reset_design(2)
set_disable_timing(2)

remove_dp_int_round

Removes the rounding attribute from inferred multiplier and multiplier-based instantiated DesignWare cells.

SYNTAX

```
status remove_dp_int_round  
[cell_list | -all]
```

Data Types

cell_list list

ARGUMENTS

cell_list

Lists the cells from which the rounding attributes will be removed.

-all

Removes rounding attributes from all cells.

DESCRIPTION

This command removes the external and internal rounding attributes that are set by the **set_dp_int_round** command.

Note that when a design is compiled and has been internally rounded, constant propagation might occur in the design. If you remove the internal rounding attribute and compile again, the final netlist might fail the Formality check as constant propagation might disconnect some nets. It is not allowed to remove the internal rounding attribute on a design that has been compiled with internal rounding feature.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the external rounding value on cell *mult*:

```
remove_dp_int_round
```

```
prompt> remove_dp_int_round [find cell mult*]
```

The following example removes the external rounding value on all cells:

```
prompt> remove_dp_int_round -all
```

SEE ALSO

`set_dp_int_round(2)`

remove_driving_cell

Removes driving-cell attributes from the specified input or inout ports of the current design.

SYNTAX

```
status remove_driving_cell  
[ports]
```

Data Types

ports collection

ARGUMENTS

ports

Specifies the input or inout ports in the current design from which to remove the driving-cell attributes.

DESCRIPTION

This command removes attributes associated with an external driving cell from the specified input or inout ports in the current design.

To view drive information on ports, use the **report_port -drive** command.

The **characterize** command automatically sets driving-cell attributes on subdesign ports based on their context in the entire design.

Note that the **remove_driving_cell** command also removes any corresponding rise or fall drive resistance attributes from the **set_drive** command from the specified ports.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes driving-cell attributes from all input and inout ports:

```
prompt> remove_driving_cell [all_inputs]
```

SEE ALSO

- all_inputs(2)
- characterize(2)
- current_design(2)
- report_port(2)
- reset_design(2)
- set_drive(2)
- set_driving_cell(2)

remove_from_collection

Removes objects from a collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

```
collection remove_from_collection
[-intersect]
collection1
object_spec
```

Data Types

```
collection1  collection
object_spec list
```

ARGUMENTS

-intersect

Removes objects from *collection1* not found in *object_spec*. Without this option, removes objects from *collection1* that are found in *object_spec*.

collection1

Specifies the base collection to be copied to the result collection. Objects matching *object_spec* are removed from the result collection.

object_spec

Specifies a list of named objects or collections to remove. The object class of each element in this list must be the same as in the base collection. If the name matches an existing collection, the collection is used. Otherwise, the objects are searched for in the database using the object class of the base collection.

DESCRIPTION

The **remove_from_collection** command removes elements from a collection, creating a new collection.

If the base collection is homogeneous, any element of the *object_spec* that is not a collection is searched for in the database using the object class of the base collection. If the base collection is heterogeneous, any element of the *object_spec* that is not a collection is ignored.

If the *-intersect* option is not specified, which is the default mode, and if nothing matches the *object_spec*, the resulting collection is a

copy of the base collection. If everything in the *collection1* option matches the *object_spec*, the result is the empty collection. With the *-intersect* option the results are reversed.

For background on collections and querying of objects, see the **collections** man page.

EXAMPLES

The following example from PrimeTime gets all input ports except "CLOCK".

```
prompt> set cPorts [remove_from_collection [all_inputs] CLOCK]
{in1 in2}
```

SEE ALSO

add_to_collection(2)
collections(2)

remove_from_rp_group

Removes a cell, relative placement group, or keepout from the specified relative placement groups.

SYNTAX

```
status remove_from_rp_group
  rp_groups
  -leaf cell_name
  -hierarchy group_name
  [-instance instance_name]
  -keepout keepout_name
```

Data Types

<i>rp_groups</i>	list or collection
<i>cell_name</i>	string
<i>group_name</i>	string
<i>instance_name</i>	string
<i>keepout_name</i>	string

ARGUMENTS

rp_groups

Specifies the relative placement groups from which to remove an item. The groups must all be in the same design.

If you do not specify this argument, no items are removed.

-leaf *cell_name*

Specifies the name of a leaf cell to remove from the specified relative placement groups. This option cannot be used with any other option.

-hierarchy *group_name*

Specifies the name of the relative placement group to remove from the specified relative placement groups.

This option can be used with the **-instance** option, but it is mutually exclusive with the **-leaf** and **-keepout** options.

-instance *instance_name*

Specifies the name of the hierarchical cell that contains the instantiated relative placement group specified in the **-hierarchy** option.

-keepout *keepout_name*

Specifies the name of the keepout to remove from the specified relative placement groups. This option cannot be used with any other option.

DESCRIPTION

This command removes an item from specified relative placement groups. The item can be either a leaf cell, a relative placement group (included or instantiated), or a keepout. This command is supported only for designs that do not contain multiply-instantiated designs.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **remove_from_rp_group** command to remove a cell from an existing relative placement group:

```
prompt> get_rp_groups ripple::grp_ripple
{ripple::grp_ripple}

prompt> remove_from_rp_group ripple::grp_ripple\
-leaf carry_in_1
1
```

SEE ALSO

add_to_rp_group(2)
create_rp_group(2)
get_rp_groups(2)
remove_rp_groups(2)
write_rp_groups(2)

remove_generated_clock

Removes a generated_clock object.

SYNTAX

```
status remove_generated_clock  
-all | clock_list
```

Data Types

clock_list list

ARGUMENTS

-all

Removes all generated clocks.

clock_list

Specifies a list of generated clock names.

DESCRIPTION

This command removes the generated clocks in the design. If the generated clocks are expanded, the actual clocks are also deleted. All of the attributes set on generated clocks (**false_paths**, **multicycle_paths**, and so on) are also removed.

To create a generated clock in the design, use the **create_generated_clock** command.

To show information about clocks and generated clocks in the design, use the **report_clock** command.

The **remove_generated_clock** command also deletes generated clocks that were defined by a library cell. Once removed, the generated clock will not return automatically until you run the **reset_design** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes the generated clock named GEN1 from the design:

```
prompt> remove_generated_clock GEN1
```

The following example removes all generated clocks from the design:

```
prompt> remove_generated_clock -all
```

SEE ALSO

`create_generated_clock(2)`

`report_clock(2)`

`reset_design(2)`

remove_host_options

Removes all **-max_cores** specifications set by the **set_host_options** command.

SYNTAX

status **remove_host_options**

ARGUMENTS

The **remove_host_options** command has no options.

DESCRIPTION

The **remove_host_options** command unsets the parameters set by the **set_host_options** command.

The only such parameter is the **-max_cores** option. Running the **remove_host_options** command disables parallel execution.

EXAMPLES

The following example removes all **-max_cores** specifications set by the **set_host_options** command.

```
prompt> remove_host_options
```

SEE ALSO

report_host_options(2)
set_host_options(2)

remove_ideal_latency

Removes ideal latency information from the specified objects.

SYNTAX

```
string remove_ideal_latency  
  [-rise | -fall]  
  [-min | -max]  
  object_list | -all
```

Data Types

object_list list

ARGUMENTS

-rise | -fall

Specifies that rise or fall ideal latency should be removed. By default, ideal latency is removed for both rise and fall ideal latency. The **-rise** and **-fall** options are mutually exclusive.

-min | -max

Specifies that maximum or minimum ideal latency should be removed. By default, it is removed for both minimum and maximum delay analysis. The **-min** and **-max** options are mutually exclusive.

object_list

Specifies a list of ports or pins from which to remove ideal latency. The *object_list* argument and the **-all** option are mutually exclusive.

-all

Removes ideal network latency from all objects. The **-all** option and the *object_list* argument are mutually exclusive.

DESCRIPTION

This command removes from specified objects the user-specified ideal latency that was previously set by the **set_ideal_latency** command. See the **set_ideal_latency** man page for more details. You must either specify an *object_list* or use **-all**.

You can remove selected portions of ideal latency by specifying applicable **-rise**, **-fall**, **-min**, and **-max** options.

To list ideal latency values, use the **report_ideal_network** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes ideal latency from the output pin named Z of cell instance U1/U2/U3:

```
prompt> remove_ideal_latency {U1/U2/U3/Z}
```

The following example removes only rise ideal latency information from a port named A:

```
prompt> remove_ideal_latency -rise {A}
```

SEE ALSO

- current_design(2)
- remove_ideal_network(2)
- remove_ideal_transition(2)
- report_ideal_network(2)
- report_timing(2)
- set_auto_disable_drc_nets(2)
- set_ideal_net(2)
- set_ideal_network(2)
- set_ideal_transition(2)

remove_ideal_net

Restores the ideal nets set by the **set_ideal_net** or **set_ideal_network -no_propagate** command to their initial nonideal state from the specified nets in the current design.

SYNTAX

```
status remove_ideal_net  
      net_list
```

Data Types

net_list list

ARGUMENTS

net_list

Specifies a list of net names from which to remove the **ideal_net** attribute. These nets must be visible from the current design.

DESCRIPTION

This command removes the **ideal_net** attribute from the individual nets specified in *net_list*, or restores the ideal nets that are set by the **set_ideal_net** command or the **set_ideal_network -no_propagate** command to their initial nonideal state from the specified nets in *net_list*. Those nets must be visible from the current design.

Ideal nets are networks of nets that are free from the max_capacitance and max_fanout design rule constraints. Ideal nets are useful to reduce DRC violations caused by clock trees, because these networks usually have high max_capacitance and max_fanout violations.

To remove the **auto_disable_drc_net** attribute from nets, use **set_auto_disable_drc_nets -none**. The **reset_design** command removes all attributes from the design, including the **ideal_net** and **auto_disable_drc_net** attributes. By default, the tool still treats clock networks as ideal nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the **ideal_net** attribute from individual nets in the design named *CLOCK_GEN*:

```
prompt> current_design CLOCK_GEN
```

```
prompt> remove_ideal_net [get_nets]
```

SEE ALSO

```
reset_design(2)  
set_auto_disable_drc_nets(2)  
set_dont_touch(2)  
set_dont_touch_network(2)  
set_ideal_net(2)
```

remove_ideal_network

Removes a set of ports or pins in an ideal network in the current design. Cells and nets in the transitive fanout of the specified objects are no longer treated as ideal.

SYNTAX

```
status remove_ideal_network  
      object_list | -all
```

Data Types

object_list list

ARGUMENTS

object_list

Specifies a list of objects (pins or ports) that are sources of an ideal network. If more than one object name is specified, enclose the objects in quotes or braces ({}). The source objects can be input ports on the design or any internal pin except pins at hierarchical boundaries. These objects must be visible from the current design.

This argument and the **-all** option are mutually exclusive.

-all

Removes all ideal networks from the current design.

This option and the *object_list* argument are mutually exclusive.

DESCRIPTION

This command removes the ideal network designation from ports or pins in an ideal network.

Ideal networks, which are an extension of ideal nets, incorporate automatic propagation of the **ideal** attribute. You specify only the source ports or source pins of the network. All nets, cells, and pins on the transitive fanin of these objects are treated as ideal by the **compile** command.

The **ideal_network** attribute is automatically spread by the tool and respread as needed during optimizations. The **remove_ideal_network** command returns to normal part or all of a network that was previously marked as ideal using the **set_ideal_network** command. See the **set_ideal_network** man page for details about the propagation rules.

In addition to disabling timing update and timing optimizations, all cells in the ideal network are set with the **dont_touch** attribute.

When part or all of the network is restored to normal using **remove_ideal_network**, the original **dont_touch** status of cells is restored.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets up an ideal network on objects in the design *CLOCK_GEN* and then removes part of the network:

```
prompt> current_design CLOCK_GEN
```

```
prompt> set_ideal_network {port1 port2}
```

```
prompt> remove_ideal_network port2
```

SEE ALSO

- remove_ideal_net(2)
- report_ideal_network(2)
- reset_design(2)
- set_auto_disable_drc_nets(2)
- set_dont_touch(2)
- set_dont_touch_network(2)
- set_ideal_latency(2)
- set_ideal_net(2)
- set_ideal_network(2)
- set_ideal_transition(2)

remove_ideal_transition

Removes ideal transition information from the specified objects.

SYNTAX

```
status remove_ideal_transition  
  [-rise | -fall]  
  [-min | -max]  
  object_list | -all
```

Data Types

object_list list

ARGUMENTS

-rise | -fall

Specifies that rise or fall ideal transition should be removed. By default, ideal transition is removed for both rise and fall ideal transition. The **-rise** and **-fall** options are mutually exclusive.

-min | -max

Specifies that maximum or minimum ideal transition should be removed. By default, ideal transition is removed for both maximum and minimum ideal transition. The **-min** and **-max** options are mutually exclusive.

object_list

Specifies a list of ports or pins from which to remove ideal transition. This argument and the **-all** option are mutually exclusive.

-all

Removes ideal network transition from all objects. This option and the *object_list* argument are mutually exclusive.

DESCRIPTION

This command removes from specified objects the user-specified ideal transition that was previously set by the **set_ideal_transition** command. See the **set_ideal_transition** man page for more details.

You can remove selected portions of ideal transition by specifying applicable **-rise**, **-fall**, **-min**, and **-max** options.

To list ideal transition values, use the **report_ideal_network** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes ideal transition from the output pin named Z of cell instance U1/U2/U3:

```
prompt> remove_ideal_transition {U1/U2/U3/Z}
```

The following example removes only rise ideal transition information a port named A:

```
prompt> remove_ideal_transition -rise {A}
```

SEE ALSO

- current_design(2)
- remove_ideal_latency(2)
- remove_ideal_network(2)
- report_ideal_network(2)
- report_timing(2)
- set_auto_disable_drc_nets(2)
- set_ideal_net(2)
- set_ideal_network(2)
- set_ideal_transition(2)

remove_ignored_layers

Removes ignored routing layers for congestion analysis and RC estimation. This command is supported only in Design Compiler topographical mode.

SYNTAX

```
status remove_ignored_layers
      list_of_layers
      [-all]
      [-min_routing_layer]
      [-max_routing_layer]
```

Data Types

list_of_layers list

ARGUMENTS

list_of_layers

Specifies the routing layers that should no longer be ignored during congestion analysis and RC estimation.

-all

Specifies that no routing layers should be ignored during congestion analysis and RC estimation.

-min_routing_layer

Removes the global minimum routing layer constraint.

-max_routing_layer

Removes the global maximum routing layer constraint.

DESCRIPTION

This command removes the ignored setting from some or all routing layers, so that the layers are used during congestion analysis and RC estimation.

Note that congestion analysis and RC estimation require that at least one horizontal and one vertical layer not be ignored.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command removes the ignored setting from all routing layers so that all layers are used for congestion analysis and RC estimation. It also removes the global minimum and maximum routing layer constraints:

```
prompt> remove_ignored_layers -all \  
       -min_routing_layer -max_routing_layer
```

SEE ALSO

report_ignored_layers(2)
report_lib(2)
set_ignored_layers(2)

remove_input_delay

Removes input delay on the specified pins or input ports.

SYNTAX

```
status remove_input_delay
  [-clock clock [-clock_fall] [-level_sensitive]]
  [-rise] [-fall]
  [-max] [-min]
  ports_pins
```

Data Types

clock string or collection of one object
ports_pins collection

ARGUMENTS

-clock *clock*

Removes delay information relative to the specified clock edge. You can specify the clock as either a string or a collection of one object.

To remove input delay that has no relative clock, use the **remove_input_delay -clock ""** command.

By default, all input delay information is removed.

-clock_fall

Removes input delay relative to the clock falling edge.

By default, input delay relative to the clock rising edge is removed.

This option is used only with the **-clock** option.

-level_sensitive

Removes level-sensitive input delay.

By default, only nonlevel-sensitive input delay is removed.

This option is used only with the **-clock** option.

-rise

Removes rising delay on the specified ports and pins.

If you do not specify either the **-rise** or **-fall** option, both rising and falling delays are removed.

-fall

Removes falling delay on the specified ports and pins.

If you do not specify either the **-rise** or **-fall** option, both rising and falling delays are removed.

-max

Removes maximum input delays on the specified ports and pins.

If you do not specify either the **-max** or **-min** option, both maximum and minimum input delays are removed.

-min

Removes minimum input delays on the specified ports and pins.

If you do not specify either the **-max** or **-min** option, both maximum and minimum input delays are removed.

ports_pins

Specifies the ports or pins from which to remove the input delay.

DESCRIPTION

This command removes input-delay values for the specified ports and pins.

You set the input delay by using the **set_input_delay** or **characterize** command. By default, all input delay is removed from the specified ports and pins. To restrict the input-delay values that are removed, use the **-clock**, **-clock_fall**, **-min**, **-max**, **-rise**, or **-fall** options.

Use the **report_port** command to list input delays associated with ports. Use the **report_design** command to list input delays of internal pins.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes all input-delay values from all input ports in the current design:

```
prompt> remove_input_delay [all_inputs]
```

The following example removes the maximum-rise input-delay values from the IN1, IN3, and BDIR12 ports:

```
prompt> remove_input_delay -max -rise {IN1 IN3 BDIR12}
```

The following example removes all input delay relative to the falling edge of CLK2 for port IN1:

```
prompt> remove_input_delay -clock [get_clocks CLK2] -clock_fall \  
[get_ports IN1]
```

The following example removes all input delay not relative to any clock for port IN4:

```
prompt> remove_input_delay -clock "" IN4
```

SEE ALSO

- all_inputs(2)
- current_design(2)
- remove_output_delay(2)
- report_design(2)
- report_port(2)
- reset_design(2)
- set_input_delay(2)
- set_output_delay(2)

remove_isolate_ports

Removes the specified ports from the list of ports that are isolated in the current design.

SYNTAX

```
status remove_isolate_ports  
    port_list
```

Data Types

port_list list

ARGUMENTS

port_list

Specifies the ports that to remove from the list of ports that are isolated in the current design. Port isolation must have been requested for these ports using the **set_isolate_ports** command.

DESCRIPTION

The **remove_isolate_ports** command removes the specified ports from the list of ports that are isolated in the current design.

This command does not remove the isolation cell inserted by a previous **compile** command. The isolation cell will be removed by a subsequent **compile** command issued after the **remove_isolate_ports** command, if the **max_area** attribute has been set on the current design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the port named *qout* from the list of ports being isolated:

```
prompt> remove_isolate_ports qout
```

SEE ALSO

report_isolate_ports(2)
reset_design(2)
set_isolate_ports(2)
set_max_area(2)

remove_isolation_cell

Removes the specified isolation cells from the design.

SYNTAX

```
status remove_isolation_cell  
[-force]  
-object_list cells
```

Data Types

cells list

ARGUMENTS

-force

Forces deletion of ISO/ELS cells for cells marked with the **dont_touch** attribute.

-object_list *cells*

Specifies the list of cells to be removed. This option is required.

DESCRIPTION

The **remove_isolation_cell** command removes isolation cells or enables level shifters. This command can be used only on a uniquified design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all isolation cells with string ISO in their name:

```
prompt> remove_isolation_cell -obj [get_cells *ISO*]
```

SEE ALSO

`insert_isolation_cell(2)`

remove_keepout_margin

Removes keepout margin of specified type for the specified cells/lib cells in the design.

SYNTAX

```
int remove_keepout_margin  
  [-type hard | soft]  
  [-derived]  
  object_list
```

Data Types

object_list list

ARGUMENTS

-type hard | soft

Specifies the type of keepout to be removed. The valid values are **hard** and **soft**. By default, keepouts of both soft and hard types are removed for the cells or library cells.

-derived

This option is provided to remove the user specified pin count based derived keepout margin values. User defined pin count based keepout margin values are set using the -tracks_per_macro_pin and/or -max_padding_per_macro and/or -min_padding_per_macro options of **set_keepout_margin** command.

object_list

Describes the list of cells or library cells for which keepouts are to be removed. The keepouts are deleted for all cells with FIXED_PLACEMENT or library cells in this list.

DESCRIPTION

This command removes keepouts for the specified cells/library cells. If you do not use the **-type** option, keepouts of both soft and hard type are deleted for the specified cells/library cells. If this command is run to delete a keepout when none exists for a cell/library cell, a warning message is printed out and no action is taken.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **remove_keepout_margin** command.

```
prompt> report_keepout_margin MY_CELL -type hard  
Cell Name: MY_CELL  
*****  
Hard Keepout - 10.0000 10.0000 10.0000 10.0000  
prompt> remove_keepout_margin -type hard MY_CELL  
prompt> report_keepout_margin MY_CELL -type soft  
Cell Name: MY_CELL  
*****  
Soft Keepout - 10.0000 10.0000 10.0000 10.0000  
prompt> remove_keepout_margin -type hard MY_LIB_CELL  
The following example will remove all derived keepouts in the design.  
prompt> remove_keepout_margin -derived *S
```

SEE ALSO

get_attribute(2)
report_keepout_margin(2)
set_keepout_margin(2)

remove_level_shifters

Removes all of the level shifters from the design.

SYNTAX

```
status remove_level_shifters  
[-force]
```

ARGUMENTS

-force

Removes level shifters even if they are set with the **dont_touch** attribute, or if they were instantiated in the original netlist.

DESCRIPTION

This command removes all level shifters from the design, except those that you explicitly marked as **dont_touch** and those that you instantiated in the original netlist. To remove level shifters that have these restrictions, use the **-force** option.

This command provides the capability to remove pre-existing or tool-inserted level shifters, allowing you to perform custom voltage adjustments by manually inserting different level-shifter cells.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command removes level shifters in a design:

```
prompt> remove_level_shifters
```

SEE ALSO

remove_libcell_subset

Removes target library family constraints from specified cells or removes a target library family defined by the **define_libcell_subset** command.

SYNTAX

```
status remove_libcell_subset  
  [-object_list cells]  
  [-family_name name]
```

Data Types

```
cells    list  
name     string
```

ARGUMENTS

-object_list *cells*

Specifies the cells from which to remove the target library family. The cells must be either sequential cells or instantiated combinational cells. You must specify either **-object_list** or **-family_name**. If neither **-object_list** nor **-family_name** is specified, an error occurs and the command quits.

-family_name *name*

Specifies the target library family to be removed. You must specify either **-family_name** or **-object_list**. If neither **-family_name** nor **-object_list** is specified, an error occurs and the command quits.

DESCRIPTION

This command removes the target library family specified by the **set_libcell_subset** command from the cells to be optimized, or it removes a target library family defined by the **define_libcell_subset** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the target library family from the u1 sequential cell:

```
prompt> remove_libcell_subset -object_list [get_cells u1]
```

The following example removes the special_flops target library family:

```
prompt> remove_libcell_subset -family_name special_flops
```

SEE ALSO

```
remove_libcell_subset(2)  
report_libcell_subset(2)  
set_libcell_subset(2)  
define_libcell_subset(2)
```

remove_license

Removes a licensed feature.

SYNTAX

```
status remove_license
      feature_list
      [-keep num_licenses]
```

Data Types

```
feature_list  list
num_licenses integer
```

ARGUMENTS

feature_list

Specifies the list of features to remove. If you specify more than one feature, the list must be enclosed in braces ({}).

By looking at your key file, you can determine all of the features licensed at your site.

-keep *num_licenses*

Specifies the number of licenses to be retained for each feature after the command has completed. If this option is not specified, all of the features' licenses are checked in.

DESCRIPTION

This command removes the specified licensed features from the features you are currently using.

For multicore runs, where multiple licenses might be required for certain features, you can use the **-keep** option to restrict the number of licenses that are checked in. This is analogous to the **get_license -quantity** option.

The **list_licenses** command provides a list of the features that you are currently using.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the multivoltage license:

```
prompt> remove_license Galaxy-MV
```

The following example removes some, but not all, of the licenses required for a multicore **compile_ultra** run in Design Compiler:

```
prompt> list_licenses
```

Licenses in use:

- DC-Expert (4)
- DC-Ultra-Features (4)
- DC-Ultra-Opt (4)
- Design-Compiler

1

```
prompt> remove_license -keep 2 {DC-Expert DC-Ultra-Opt DC-Ultra-Features}
```

1

```
prompt> list_licenses
```

Licenses in use:

- DC-Expert (2)
- DC-Ultra-Features (2)
- DC-Ultra-Opt (2)
- Design-Compiler

1

SEE ALSO

- check_license(2)
- license_users(2)
- list_licenses(2)
- get_license(2)

remove_link_library_subset

Removes the link library subset constraint from the top-level design or from the specified instances.

SYNTAX

```
status remove_link_library_subset  
[-object_list cells]  
[-top]
```

ARGUMENTS

-object_list *cells*

Specifies the cells from which to remove the link library subset. The cells must be instances of hierarchical designs, macros, or pads. You must specify **-top**, **-object_list**, or both options.

-top

Removes the link library subset from the top-level design. You must specify **-top**, **-object_list**, or both options.

DESCRIPTION

This command removes the link library subset from the specified instances or the top-level design. The subset was specified previously by the **set_link_library_subset** command.

Multicorner-Multimode Support

This command applies to the current scenario only. The command is scenario-specific, so it must be specified in each scenario where you want it to take effect.

EXAMPLES

The following example removes the link library subset from the top-level design and from the u1 instance:

```
prompt> remove_link_library_subset -object_list [get_cells u1] -top
```

SEE ALSO

report_link_library_subset(2)
set_link_library_subset(2)
link_library(3)

remove_min_pulse_width

Removes a previously specified minimum pulse width constraint from specified clocks or clock pins.

SYNTAX

```
status remove_min_pulse_width  
  [-low]  
  [-high]  
  [object_list]
```

Data Types

object_list list

ARGUMENTS

-low

Removes the minimum pulse width constraint only for low clock pulses (high-low-high).

-high

Removes the minimum pulse width constraint only for high clock pulses (low-high-low).

If you do not specify the *-low* or *-high* option, the command removes the minimum pulse width constraint for all types of clock pulses.

object_list

Specifies a list of clocks or clock pins in the current design from which to remove the minimum pulse width constraint. If you specify a clock, all clock pins connected to the clock have their minimum pulse width constraints removed. If you do not specify any objects, the minimum pulse width check is removed from all objects in the current design.

DESCRIPTION

The **remove_min_pulse_width** command removes pulse width checks previously specified by the **set_min_pulse_width** command for clock signals at clock pins of sequential devices. This command removes the constraints in all scenarios.

To generate a report of pulse width constraints, use the **report_constraint** or **report_min_pulse_width** command.

EXAMPLES

The following example removes a minimum pulse width requirement previously set for clock CK1.

```
prompt> remove_min_pulse_width [get_clocks CK1]
```

The following example removes a minimum pulse width requirement previously set for pin reg1/CK.

```
prompt> remove_min_pulse_width reg1/CK
```

SEE ALSO

```
report_constraint(2)  
report_min_pulse_width(2)  
set_min_pulse_width(2)
```

remove_multibit

Removes the multibit components from the current design. Removes or detaches cells from the multibit components in a design.

SYNTAX

```
status remove_multibit  
      object_list
```

Data Types

object_list list

ARGUMENTS

object_list

Specifies a list of multibit components, cells, designs and cell instances to remove or detach. The objects must be from the current design. This argument is required.

DESCRIPTION

This command deletes multibit components from the design and detaches cells or cell instances from multibit components.

Specify multibit components in the *object_list* to remove the component from the design. Specify cells or cell instances in the list to detach the cell from the multibit component in which the cell is contained. Specify design objects to remove all the multibit components of the given design.

When design objects are specified, multibit components are removed only from the scope of the specified design. This command doesn't recursively remove multibit components from the sub designs within the specified design.

All the cell instances that were detached from their multibit components will be marked with the `remove_multibit` attribute.

This command does not delete cells from the design; it only manipulates multibit components in the design.

This command requires that the design be linked. If an attempt to link the design fails, the command is terminated.

EXAMPLES

The following example shows the multibit component status on the current design before any changes are made:

prompt> **report_multibit**

```
*****
Report : multibit
Design : subtest
Version: 2007.02
Date   : Fri Aug 29 10:01:19 2007
*****
```

Attributes:

b - black box (unknown)
h - hierarchical
n - noncombinational
r - removable
u - contains unmapped logic

Multibit Component : y_reg

Cell	Reference	Library	Area	Width	Attributes
y_reg[3]	**SEQGEN**		0.00	1	n, u
y_reg[2]	**SEQGEN**		0.00	1	n, u
y_reg[1]	**SEQGEN**		0.00	1	n, u
y_reg[0]	**SEQGEN**		0.00	1	n, u
Total 4 cells			0.00	4	

Multibit Component : z_reg

Cell	Reference	Library	Area	Width	Attributes
z_reg[3]	**SEQGEN**		0.00	1	n, u
z_reg[2]	**SEQGEN**		0.00	1	n, u
z_reg[1]	**SEQGEN**		0.00	1	n, u
z_reg[0]	**SEQGEN**		0.00	1	n, u
Total 4 cells			0.00	4	

Total 2 Multibit Components

This example first removes all multibit components and then shows the updated report:

prompt> **remove_multibit ""**

1

prompt> **report_multibit**

```
*****
Report : multibit
Design : subtest
Version: 2007.02
Date   : Fri Aug 29 10:01:19 2007
*****
```

Attributes:

b - black box (unknown)

h - hierarchical
 n - noncombinational
 r - removable
 u - contains unmapped logic

Total 0 Multibit Components

This example first removes the specified cell from the multibit component. The component remains with its width reduced by 1. The example then shows the updated report:

prompt> **remove_multibit z_reg[2]**

prompt> **report_multibit**

Report : multibit
 Design : subtest
 Version: 2007.02
 Date : Fri Aug 29 10:01:19 2007

Attributes:

b - black box (unknown)
 h - hierarchical
 n - noncombinational
 r - removable
 u - contains unmapped logic

Multibit Component : y_reg

Cell	Reference	Library	Area	Width	Attributes
y_reg[3]	**SEQGEN**		0.00	1	n, u
y_reg[2]	**SEQGEN**		0.00	1	n, u
y_reg[1]	**SEQGEN**		0.00	1	n, u
y_reg[0]	**SEQGEN**		0.00	1	n, u
Total 4 cells			0.00	4	

Multibit Component : z_reg

Cell	Reference	Library	Area	Width	Attributes
z_reg[3]	**SEQGEN**		0.00	1	n, u
z_reg[1]	**SEQGEN**		0.00	1	n, u
z_reg[0]	**SEQGEN**		0.00	1	n, u
Total 3 cells			0.00	3	

Total 2 Multibit Components

This example runs the command on a cell instance. The instance named *U1/U2* is a unique instantiation of the design named *BOT*.

prompt> **remove_multibit U1/U2/some_reg**
 1

This example runs the command on a multibit component from a subdesign instance. The instance named *U1/U2* is a unique instantiation of the design named *BOT*.

```
prompt> remove_multibit U1/U2/mult_comp  
1
```

SEE ALSO

- `create_multibit(2)`
- `get_multibits(2)`
- `report_compile_options(2)`
- `report_multibit(2)`
- `set_multibit_options(2)`

remove_net

Removes nets from the current design.

SYNTAX

```
status remove_net  
  net_list | -all  
  [-only_physical]
```

Data Types

net_list list

ARGUMENTS

net_list

Specifies a list of nets to remove from the current design. Each net name must exist in the current design. You must specify either *net_list* or **-all**.

-all

Removes all nets in the current design. You must specify either **-all** or *net_list*.

-only_physical

Removes only physical nets in the current design. The physical nets are those nets that do not connect to a logical netlist.

DESCRIPTION

This command removes nets or net instances from the current design. Net connections to pins or ports are disconnected.

You cannot remove bused nets with the **remove_net** command. Use the **remove_bus** command to remove bused nets. Scalar (single bit) nets that are components of a bused net cannot be removed. You must remove bused nets first.

To create nets, use the **create_net** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the nets in the current design:

```
prompt> get_nets ""  
{NET0 NET1 NET2 NET3 MY_CHECK PARITY}
```

```
prompt> remove_net "N*"  
Removing net 'NET0' in design 'my_design'.  
Removing net 'NET1' in design 'my_design'.  
Removing net 'NET2' in design 'my_design'.  
Removing net 'NET3' in design 'my_design'.
```

```
prompt> get_nets ""  
{MY_CHECK PARITY}
```

The following example removes all remaining nets in the current design:

```
prompt> remove_net -all  
Removing net 'MY_CHECK' in design 'my_design'.  
Removing net 'PARITY' in design 'my_design'.
```

```
prompt> get_nets ""  
{}
```

The following example removes the physical nets:

```
prompt> remove_net -only_physical {physnet1 physnet2}
```

The following example removes a net instance:

```
prompt> remove_net {U1/MY_NET U2/MY_NET}  
Removing net 'U1' in design 'mydesign_1'.  
Removing net 'U2' in design 'mydesign_2'.
```

SEE ALSO

`create_net(2)`
`current_design(2)`
`remove_bus(2)`

remove_net_routing_layer_constraints

Removes the routing layer constraints for the specified nets.

SYNTAX

status **remove_net_routing_layer_constraints**
list_of_nets

Data Types

list_of_nets collection

ARGUMENTS

list_of_nets

Specifies the nets for which to remove the routing layer constraints.

DESCRIPTION

This command removes the routing layer constraints for the specified nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the routing layer constraints for a net named netA.

```
prompt> remove_net_routing_layer_constraints {netA} \
```

SEE ALSO


```
report_net_routing_layer_constraints(2)  
set_net_routing_layer_constraints(2)
```

remove_net_search_pattern

Removes a net search pattern or all net search patterns.

SYNTAX

```
status remove_net_search_pattern  
-pattern id | -all
```

Data Types

id integer

ARGUMENTS

-pattern *id*

Specifies a pattern ID that you want to remove. When a pattern ID is removed, the corresponding delay estimation options specified for the search pattern by the **set_net_search_pattern_delay_estimation_options** command are also removed.

-all

Eliminates all search patterns in a design. When you specify the **-all** option, all delay estimation options that were specified by the **set_net_search_pattern_delay_estimation_options** command are removed.

DESCRIPTION

The **remove_net_search_pattern** command will remove either a specified net search pattern (using **-pattern**) or all net search patterns (using **-all**).

Once a pattern ID is removed, it cannot be reused. For example, if you have 10 existing patterns and you remove pattern ID 7 by using the **remove_net_search_pattern -pattern 7** command, even though pattern ID 7 is now empty, the next time you create a net search pattern, the pattern ID will be 11.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

Here are some examples of how you can use the **remove_net_search_pattern** command:

```
prompt> remove_net_search_pattern -pattern 7
```

```
prompt> remove_net_search_pattern -all
```

SEE ALSO

`create_net_search_pattern(2)`

`report_net_search_pattern(2)`

`set_net_search_pattern_delay_estimation_options(2)`

`report_net_search_pattern_delay_estimation_options(2)`

`set_net_search_pattern_priority(2)`

`report_net_search_pattern_priority(2)`

`get_matching_nets_for_pattern(2)`

remove_net_shape

Removes all net shapes in the current design.

SYNTAX

```
status remove_net_shape  
    net_shapes  
    [-verbose]
```

ARGUMENTS

The **remove_net_shape** command has no arguments.

DESCRIPTION

This command removes all net shapes in the current design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all net shapes in the current design. The current design has only two net shapes:

```
prompt> remove_net_shape  
Removed net shape HWIRE#7170.  
Removed net shape HWIRE#7171.  
1
```

SEE ALSO

[create_net_shape\(2\)](#)

remove_user_shape(2)

remove_ocvm

Removes parametric on-chip variation (POCV) derating previously set with the **read_ocvm** command.

SYNTAX

```
status remove_ocvm  
  [-coefficient]  
  [-derate]  
  [object_list]
```

Data Types

object_list list

ARGUMENTS

-coefficient

This option is not used in the Design Compiler tool.

-derate

This option is not used in the Design Compiler tool.

object_list

Causes the POCV coefficients set on the cells or library cells in the *object_list* to be removed.

DESCRIPTION

This command removes user-specified parametric on-chip variation (POCV) coefficients that were previously set by the **read_ocvm** command. If the command is specified with no options, all POCV coefficients are removed.

To report the POCV derating information that has been set, use the **report_ocvm** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes the random advanced OCV coefficients from the library cell named *lib/bufA*:

```
prompt> remove_ocvm -coefficient lib/bufA
```

The following example removes all advanced OCV derate factors that were set using the **read_ocvm** command:

```
prompt> remove_ocvm -derate
```

The following example removes all advanced OCV derate factors from the current design:

```
prompt> remove_ocvm -derate [current_design]
```

SEE ALSO

- `read_ocvm(2)`
- `remove_ocvm(2)`
- `report_ocvm(2)`
- `timing_pocvm_corner_sigma(3)`
- `timing_pocvm_enable_analysis(3)`

remove_output_delay

Removes output delay on pins or output ports.

SYNTAX

```
status remove_output_delay
  [-clock clock [-clock_fall] [-level_sensitive]]
  [-rise]
  [-fall]
  [-max]
  [-min]
  port_pin_list
```

Data Types

clock string or collection of one object
port_pin_list list

ARGUMENTS

-clock *clock*

Removes delay information relative to the specified clock edge. If **-clock** is not specified, all output delay is removed. To remove output delay that has no relative clock, use **remove_output_delay -clock ""**.

You can specify the clock as either a string or a collection of one object.

-clock_fall

Removes output delay relative to the clock falling edge. If you do not specify **-clock_fall**, output delay relative to the clock rising edge is removed.

This option is only used with the **-clock** option.

-level_sensitive

Removes level-sensitive output delay for the specified clock. If this option is not used, nonlevel-sensitive output delay is removed.

-rise

Removes rising delay on *port_pin_list*. If you do not specify **-rise** or **-fall**, both rising and falling delays are removed.

-fall

Removes falling delay on *port_pin_list*. If you do not specify **-rise** or **-fall**, both rising and falling delays are removed.

-max

Removes maximum output delays on *port_pin_list*. If you do not specify **-max** or **-min**, both maximum and minimum output delays are removed.

-min

Removes minimum output delays on *port_pin_list*. If you do not specify **-max** or **-min**, both maximum and minimum output delays are removed.

port_pin_list

Specifies a list of port or pin names in the current design. Output delay is removed from each object in *port_pin_list*. To specify more than one object, enclose the objects in quotes (") or braces ({}).

DESCRIPTION

This command removes output delay values for objects in the current design. Output delay is set by the **set_output_delay** or the **characterize** command. By default, all output delay on each object in *port_pin_list* is removed. To restrict the removed output delay values, use the **-clock**, **-clock_fall**, **-min**, **-max**, **-rise**, or **-fall** option.

To list output delays associated with ports, use the **report_port** command.

To list output delays of internal pins, use the **report_design** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes all output-delay values from all output ports in the current design:

```
prompt> remove_output_delay [all_outputs]
```

This example removes output maximum-rise delay values from the OUT1, OUT3, and BIDIR12 ports:

```
prompt> remove_output_delay -max -rise {OUT1 OUT3 BIDIR12}
```

This example removes all output delays for port OUT1, relative to the falling edge of CLK2:

```
prompt> remove_output_delay -clock [get_clocks CLK2] \  
-clock_fall [get_ports OUT1]
```

This example removes all output-delay values not relative to any clock for the OUT2 port:

```
prompt> remove_output_delay -clock "" OUT2
```

SEE ALSO

[all_outputs\(2\)](#)

```
current_design(2)
remove_input_delay(2)
report_design(2)
report_port(2)
reset_design(2)
set_input_delay(2)
set_output_delay(2)
```

remove_pass_directories

Removes the data directories associated with the specified passes.

SYNTAX

```
status remove_pass_directories  
    pass_list
```

Data Types

pass_list list

ARGUMENTS

pass_list

Specifies a list of directory names that list the passes whose data directories are removed.

DESCRIPTION

This command removes the data directories associated with the specified passes. This command deletes the specified directories under the Automated Chip Synthesis working directory (specified by the **acs_work_dir** variable) for each specified pass.

SEE ALSO

`create_pass_directories(2)`
`acs_work_dir(3)`

remove_path_group

Removes a list of path groups from the current design.

SYNTAX

```
string remove_path_group  
      group_list
```

Data Types

group_list list

ARGUMENTS

group_list

Specifies the path groups to remove.

DESCRIPTION

The **remove_path_group** command removes a list of path groups from the current design.

Path groups control the optimization cost function and timing reports. You can see what path groups remain in the design by using the **get_path_groups** command.

To create other path groups, use the **group_path** command.

EXAMPLES

The command set below first creates two path groups named *grp1* and *grp2*, and then removes the groups:

```
prompt> group_path -name grp1 -to data_out  
1
```

```
prompt> group_path -name grp2 -from input3 -to data_out  
1
```

```
prompt> remove_path_group {grp1 grp2}
```

1

SEE ALSO

group_path(2)

report_path_group(2)

remove_pin_guides

Removes all pin guides from the design.

SYNTAX

```
status remove_pin_guides  
-all | patterns  
[-verbose]
```

ARGUMENTS

-all

Removes all pin guides.

-verbose

Enables verbose output.

DESCRIPTION

This command deletes the pin guides from the design. Design Compiler in topographical mode does not support the removal of pin guides by name. Use the **-all** option to remove all pin guides in the design, or use the following command:

```
prompt> remove_pin_guides *
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all pin guides:

```
prompt> remove_pin_guides -all
```

SEE ALSO

remove_pin_map

Removes a design port-to-package pin mapping for a boundary-scan design.

SYNTAX

```
status remove_pin_map  
      package_name
```

Data Types

```
package_name  string
```

ARGUMENTS

package_name

Specifies the name of the package in a port-to-pin mapping file read in for the design. The package name must match one of the package names specified in a port-to-pin mapping file previously read using the **read_pin_map** command.

DESCRIPTION

The **remove_pin_map** command allows you to delete a port-to-pin mapping for a boundary-scan design read in through the **read_pin_map** command.

Use the **report_packages** command to list the package names of the port-to-pin mappings files that you read in for the current design.

EXAMPLES

The following example deletes a design port-to-package pin mapping named *test_pkg* for a boundary-scan design:

```
prompt> remove_pin_map test_pkg
```

SEE ALSO

get_attribute(2)
read_pin_map(2)

remove_pin_name_synonym

Removes pin name synonym definitions.

SYNTAX

```
status remove_pin_name_synonym  
[-all]  
[synonym_list]
```

Data Types

synonym_list list

ARGUMENTS

-all

Removes all pin name synonym definitions. You must specify either the **-all** option or the *synonym_list* argument. The option and argument are mutually exclusive.

synonym_list

Removes pin name synonym definitions that match one of the pin name synonym strings in the list. You must specify either the **-all** option or the *synonym_list* argument. The option and argument are mutually exclusive.

DESCRIPTION

The **remove_pin_name_synonym** command deletes some or all pin name synonym definitions.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples set and remove pin name synonyms:

```
prompt> set_pin_name_synonym SYN_CD CD
```

1

```
prompt> set_pin_name_synonym SYN_QN QN
```

1

```
prompt> set_pin_name_synonym \
```

```
-full_name mid1/bot1/LSR0P_at_bot/SYN_R mid1/bot1/LSR0P_at_bot/R
```

1

```
prompt> set_pin_name_synonym \
```

```
-full_name this/is/a/synonym mid1/FJK2SP_at_mid/TI
```

1

```
prompt> set_pin_name_synonym \
```

```
-full_name mid1/FJK2SP_at_mid/J mid2/bot2/LSR0P_at_bot/Q
```

1

```
prompt> report_pin_name_synonym
```

```
*****
```

```
Report : pin name synonym
```

```
Design : top
```

```
Version: X-2005.09
```

```
Date : Wed Jun 29 10:29:04 2005
```

```
*****
```

Synonym	Pin Name	Type

this/is/a/synonym	mid1/FJK2SP_at_mid/TI	full name
mid1/FJK2SP_at_mid/J	mid2/bot2/LSR0P_at_bot/Q	full name
mid1/bot1/LSR0P_at_bot/SYN_R	mid1/bot1/LSR0P_at_bot/R	full name
SYN_QN	QN	simple name
SYN_CD	CD	simple name

1

```
prompt> remove_pin_name_synonym SYN_QN
```

1

```
prompt> remove_pin_name_synonym this/is/a/synonym
```

1

```
prompt> report_pin_name_synonym
```

```
*****
```

```
Report : pin name synonym
```

```
Design : top
```

```
Version: X-2005.09
```

```
Date : Wed Jun 29 10:29:04 2005
```

```
*****
```

Synonym	Pin Name	Type

mid1/FJK2SP_at_mid/J	mid2/bot2/LSR0P_at_bot/Q	full name
mid1/bot1/LSR0P_at_bot/SYN_R	mid1/bot1/LSR0P_at_bot/R	full name
SYN_CD	CD	simple name

```
-----  
1  
  
prompt> remove_pin_name_synonym -all  
1  
  
prompt> report_pin_name_synonym  
  
*****  
Report : pin name synonym  
Design : top  
Version: X-2005.09  
Date   : Wed Jun 29 10:29:04 2005  
*****  
  
No pin name synonym  
1
```

SEE ALSO

report_pin_name_synonym(2)
set_pin_name_synonym(2)

remove_placement_blockage

Removes placement blockages.

SYNTAX

```
status remove_placement_blockage  
  patterns | -name name | -all  
  [-verbose]
```

Data Types

<i>patterns</i>	collection
<i>name</i>	string

ARGUMENTS

patterns

Specifies the pattern by which this command finds and removes a placement blockage. The patterns can be a collection of blockages, or other formats, such as * or pb#* (removes all placement blockages) or pb#7789 (removes one placement blockage whose *object_id* is 7789).

The *patterns* argument, the **-name** option, and the **-all** option are mutually exclusive.

-name *name*

Specifies a name by which this command finds and removes a placement blockage. The name must be the same name specified for the placement blockage when it was created with the **create_placement_blockage** command. The name must not be a runtime name, such as pb#7789.

The *patterns* argument, the **-name** option, and the **-all** option are mutually exclusive.

-all

Removes all placement blockages in the current design.

The *patterns* argument, the **-name** option, and the **-all** option are mutually exclusive.

-verbose

Prints additional messages.

DESCRIPTION

This command removes all specified placement blockages.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all placement blockages:

```
prompt> remove_placement_blockage -all
```

The following example uses the *patterns* argument to remove placement blockages:

```
prompt> remove_placement_blockage \  
       [get_placement_blockages * -within {{2 2} {25 25}}]
```

SEE ALSO

`create_placement_blockage(2)`

`create_route_guide(2)`

`remove_route_guide(2)`

remove_port

Removes ports from the current design or its subdesign.

SYNTAX

```
status remove_port  
    port_list
```

Data Types

port_list list

ARGUMENTS

port_list

Specifies a list of ports to be removed from the current design. Each port name must exist in the current design.

DESCRIPTION

This command removes ports from the current design or its subdesign. Bused ports cannot be removed with **remove_port**; use **remove_bus** to remove bused ports. Also, scalar (single bit) ports that are components of a bused port cannot be removed. In this case, you must first remove the bused port.

To create ports, use the **create_port** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **remove_port** to remove ports from the current design:

```
prompt> get_ports ""  
{A1 A2, B1, B2, OUT1}
```

```
prompt> remove_port "B*"
```

```
Removing port 'B1' in design 'my_design'.  
Removing port 'B2' in design 'my_design'.  
1
```

```
prompt> get_ports ""  
{A1 A2 OUT1}
```

In the following example, ports are removed from the subdesign *MID*. *U1* is an instance of the design *MID*.

```
prompt> remove_port [get_ports U1/p1 U1/p2]  
Removing port 'U1/p1' in design 'MID'.  
Removing port 'U1/p2' in design 'my_design'.  
1
```

SEE ALSO

```
create_port(2)  
current_design(2)  
remove_bus(2)  
remove_unconnected_ports(2)
```

remove_power_domain

Removes the specified power domains.

SYNTAX

```
status remove_power_domain  
    power_domains | -all
```

Data Types

power_domains collection

ARGUMENTS

power_domains

Specifies the power domains to be deleted. The name must be a simple (non-hierarchical) name.

This option is mutually exclusive with the **-all** option.

-all

Deletes all power domains in the current design. This option is mutually exclusive with the *power_domains* argument.

DESCRIPTION

The **remove_power_domain** command removes existing power domains. Before you can delete a power domain, you must remove all relationships between the power domain and any supply ports, supply nets, power switches, or child power domains.

The command fails if it cannot find any specified power domains or if there are any supply ports, supply nets, or power switches associated with the specified power domains.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the PD1 power domain:

```
prompt> remove_power_domain PD1  
1
```

The following example removes power domains whose name start with PD:

```
prompt> remove_power_domain [get_power_domains -hierarchical PD*]  
1
```

SEE ALSO

- `create_power_domain(2)`
- `get_power_domains(2)`
- `report_power_domain(2)`
- `set_scope(2)`

remove_preferred_routing_direction

Removes the preferred routing direction for the given routing layers.

SYNTAX

```
string remove_preferred_routing_direction  
-layers list_of_layers
```

ARGUMENTS

-layers *list_of_layers*

Specifies the list or collection of layers for which the preferred routing directions are to be removed.

DESCRIPTION

This command removes the preferred routing direction for the routing layers specified from the design and the preferred routing direction for those routing layers that default to the one specified in the reference library.

This command issues a warning for layers that do not have a defined preferred routing direction in the reference library, since at least one preferred routing direction setting must exist for each layer.

The command issues appropriate warnings after the command is run when adjacent layers have the same routing direction after the execution of the command.

The command removes the setting from the design, and not the reference library. The change is to the persistent data and is reflected in the design database when the design is saved.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the preferred routing direction for layers *m1* and *m2*:

```
prompt> remove_preferred_routing_direction -layers {m5 m6}
```

SEE ALSO

report_preferred_routing_direction(2)
set_preferred_routing_direction(2)

remove_propagated_clock

Removes propagated clock latency previously set with the **set_propagated_clock** command, which restores ideal clock latency for the specified objects.

SYNTAX

```
status remove_propagated_clock  
-all | object_list
```

Data Types

object_list collection

ARGUMENTS

-all

Removes propagated clocking from all objects in the current scenario.

object_list

Specifies the clocks, ports, or pins from which to remove the propagated clock specification.

DESCRIPTION

This command removes propagated clocking previously set by the **set_propagated_clock** command from all objects or from specified clocks, ports, or pins in the current scenario. This restores ideal clocking for the specified objects.

Ideal clocking means clock networks have a specified latency (from the **set_clock_latency** command), or zero latency by default. Latency is the amount of time a clock signal takes to be propagated from the ideal waveform origin point to the clock pin of the sequential device.

Propagated clock latency is used for after final clock tree generation and layout. Ideal clock latency specifies a prelayout estimate of the clock tree delay.

You can override the propagated clock specification for an object by specifying an ideal latency for the object with the **set_clock_latency** command.

To report propagated clock attributes on clocks, use the **report_clock -skew** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes propagated clock specification set on all clocks in the current scenario:

```
prompt> remove_propagated_clock [all_clocks]
```

To remove propagated clock specification set on all clocks, ports, and pins in the current scenario, use the **remove_propagated_clock -all** command.

SEE ALSO

- `create_clock(2)`
- `remove_attribute(2)`
- `report_clock(2)`
- `reset_design(2)`
- `set_clock_latency(2)`
- `set_propagated_clock(2)`

remove_route_guide

Removes the specified route guides.

SYNTAX

```
status remove_route_guide
[-verbose]
-name route_guide_name | -all | patterns
```

Data Types

```
route_guide_name  string
patterns          collection
```

ARGUMENTS

-verbose

Prints additional messages.

-name *route_guide_name*

Specifies the name of the route guide to remove.

The name must be the same name you used when you created the route guide with the **create_route_guide** command. Do not use a runtime name, such as "rg#7689".

This option is mutually exclusive with the *patterns* argument and the **-all** option, but you must specify one.

-all

This option removes all route guides in the current design.

This option is mutually exclusive with the **-name** option and the *patterns* argument, but you must specify one.

patterns

Specifies the route guides to remove. The patterns can be a collection of route guides or other formats, such as the following:

- ** or rg** removes all route guides
- *rg#7789* removes one route guide whose object_id is 7789

This argument is mutually exclusive with the **-name** and **-all** options, but you must specify one.

DESCRIPTION

The **remove_route_guide** command removes the specified route guides.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the route guides specified by the **get_route_guides** command:

```
prompt> remove_route_guide [get_route_guides -within {{2 2} {25 25}}]
1
```

The following example removes all route guides:

```
prompt> remove_route_guide -all
1
```

SEE ALSO

create_placement_blockage(2)
create_route_guide(2)
get_placement_blockages(2)
remove_placement_blockage(2)

remove_routing_rules

Removes nondefault routing rules in a design defined by the **define_routing_rule** command.

SYNTAX

```
int remove_routing_rules  
  [-all]  
  rule_name_list
```

Data Types

rule_name_list list

ARGUMENTS

-all

Removes all nondefault routing rules.

rule_name_list

Removes nondefault routing rules with the specified names.

DESCRIPTION

This command removes the nondefault routing rules defined in the design. If you use the **-all** option, all nondefault routing rules in the design are removed. If you specify a list of names, the nondefault routing rules in the list are removed.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the nondefault routing rule named *wire_rule1*.

```
prompt> remove_routing_rules wire_rule1
```

SEE ALSO

`define_routing_rule(2)`
`report_routing_rules(2)`

remove_rp_group_options

Removes relative placement (RP) group attributes from the specified relative placement groups.

SYNTAX

```
collection remove_rp_group_options  
  rp_groups  
  [-ignore]  
  [-x_offset]  
  [-y_offset]  
  [-cell_orient_opt]  
  [-auto_blockage]  
  [-disable_buffering]  
  [-allow_non_rp_cells]  
  [-ignore_rows]  
  [-max_rp_width]  
  [-max_rp_height]
```

Data Types

rp_groups list or collection

ARGUMENTS

rp_groups

Specifies the relative placement groups from which to remove the specified relative placement group attributes.

-ignore

Removes the **-ignore** attribute, which indicates that the tool is to ignore this relative placement group during placement and not treat it as a relative placement group.

-x_offset

Removes the **-x_offset** attribute of the relative placement group.

-y_offset

Removes the **-y_offset** attribute of the relative placement group.

-cell_orient_opt

Removes the **-cell_orient_opt** attribute of the relative placement group.

-auto_blockage

Removes the **-auto_blockage** attribute of the relative placement group.

-disable_buffering

Removes the **-disable_buffering** attribute of the relative placement group.

-allow_non_rp_cells

Removes the **-allow_non_rp_cells** attribute of the relative placement group.

-ignore_rows

Removes the **-ignore_rows** attribute of the relative placement group.

-max_rp_width

Removes the **-max_rp_width** attribute of the relative placement group.

-max_rp_height

Removes the **-max_rp_height** attribute of the relative placement group.

DESCRIPTION

This command removes the attributes of the relative placement groups. This command is supported only for designs that do not contain multiply-instantiated designs. This command returns a collection containing the relative placement groups for which attributes have changed. If attributes have not changed for any group, an empty string is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes attributes from a relative placement group:

```
prompt> remove_rp_group_options ripple::grp_ripple -ignore \  
-x_offset -y_offset \  
{ripple::grp_ripple}
```

SEE ALSO

add_to_rp_group(2)
all_rp_groups(2)
create_rp_group(2)
get_rp_groups(2)
remove_from_rp_group(2)
remove_rp_groups(2)

```
set_rp_group_options(2)  
write_rp_groups(2)
```

remove_rp_groups

Removes a list of relative placement (RP) groups.

SYNTAX

```
status remove_rp_groups  
  rp_groups | -all  
  [-hierarchy]  
  [-quiet]
```

Data Types

rp_groups list or collection

ARGUMENTS

rp_groups

Specifies a list of relative placement groups to remove.

This argument and the **-all** option are mutually exclusive.

-all

Specifies that all relative placement groups will be removed.

This option cannot be used with either the *rp_groups* argument or the **-hierarchy** option.

-hierarchy

Specifies that all relative placement groups within the hierarchy of the groups in *rp_groups* will be removed. By default, subgroups are not removed.

This option cannot be used with the **-all** option.

-quiet

Turns off messages that would otherwise be issued as relative placement groups are removed.

DESCRIPTION

The **remove_rp_groups** command removes a list of relative placement groups. When a relative placement group is removed, the memory it occupies is freed, so the object is no longer in the design.

Relative placement usage is available with Design Compiler Ultra.

Note that memory is freed to be reused by this same process. However, memory is not returned to the operating system until you exit the tool.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **remove_rp_groups** to delete a relative placement group.

```
prompt> get_rp_groups  
{mul::grp_mul ripple::grp_ripple example3::top_group}
```

```
prompt> remove_rp_groups ripple::grp_ripple  
Removing rp group 'ripple::grp_ripple'  
1
```

```
prompt> get_rp_groups ripple::grp_ripple  
Error: Can't find object 'grp_ripple'. (UID-109)
```

```
prompt> remove_rp_groups -all  
Removing rp group 'mul::grp_mul'  
Removing rp group 'example3::top_group'  
1
```

SEE ALSO

add_to_rp_group(2)
create_rp_group(2)
write_rp_groups(2)

remove_rtl_load

Removes previously-set capacitance and resistance RTL load values from pins, ports, and nets.

SYNTAX

```
status remove_rtl_load  
-all | pin_net_list
```

Data Types

pin_net_list list

ARGUMENTS

-all

Specifies that all RTL loads in the current design are to be removed. You must specify either **-all** or *pin_net_list*, but not both.

pin_net_list

Specifies a list of ports, pins, and nets in the current design whose RTL loads are to be removed. You must specify either **-all** or *pin_net_list*, but not both.

DESCRIPTION

The **remove_rtl_load** command removes RTL load information, previously annotated by the **set_rtl_load** or **calculate_rtl_load** command, from the specified objects or from all objects in the current design. If a net is specified, the RTL loads are removed from all pins of the net.

The **reset_design** command removes all attributes from the current design, including all capacitance and resistance RTL load values.

EXAMPLES

The following example removes RTL load values from each of the specified pins and ports:

```
prompt> remove_rtl_load {my_port my_ram/ADDR[*]}
```


The following example removes RTL load values from the pins of the net named *net1*:

```
prompt> remove_rtl_load my_hier/net1
```

The following example removes all RTL loads from the current design:

```
prompt> remove_rtl_load -all
```

SEE ALSO

[current_design\(2\)](#)

[reset_design\(2\)](#)

[set_rtl_load\(2\)](#)

remove_scaling_lib_group

Removes any previously specified scaling library group from the current design or from a subdesign.

SYNTAX

```
status remove_scaling_lib_group
      [-object_list objects]
```

Data Types

objects list

ARGUMENTS

-object_list *objects*

Specifies the cells or top-level ports from which to remove any requested scaling library groups. If you do not use this option, groups are removed from the top-level design.

DESCRIPTION

The **remove_scaling_lib_group** command removes the scaling library group set on the design objects that was previously by **set_scaling_lib_group** commands. After this removal, the tool will not perform scaling between libraries for voltage and/or temperature.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **remove_scaling_lib_group** command:

```
prompt> remove_scaling_lib_group -object_list top/inv
```

SEE ALSO

`create_scenario(2)`
`define_scaling_lib_group(2)`
`set_operating_conditions(2)`
`set_scaling_lib_group(2)`

remove_scan_group

Removes an existing scan-group specification previously specified using the **set_scan_group** command.

SYNTAX

```
status remove_scan_group  
      scan_group_name
```

Data Types

```
scan_group_name  string
```

ARGUMENTS

scan_group_name

Specifies the name of the scan group. The scan group must be a scan group previously defined using the **set_scan_group** command.

DESCRIPTION

This command removes an existing scan-group specification previously specified using the **set_scan_group** command.

The **remove_scan_group** command cannot remove scan groups under these two circumstances:

- The scan group you want to remove has not been previously defined using the **set_scan_group** command.
- The scan group you want to remove was included in a **set_scan_path** specification.

In such cases, **remove_scan_group** exits with a warning.

EXAMPLES

The following is an example of the **remove_scan_group** command:

```
prompt> remove_scan_group my_shift_reg
```

SEE ALSO

`insert_dft(2)`
`preview_dft(2)`
`report_scan_group(2)`
`set_scan_group(2)`

remove_scan_link

Removes a scan-link specification for the current design.

SYNTAX

```
status remove_scan_link
      scan_link_name
      Wire | Lockup
      [-test_mode test_mode]
```

Data Types

```
scan_link_name  string
test_mode       string
```

ARGUMENTS

scan_link_name

Specifies the name of the scan link to be removed from the specification.

{Wire | Lockup}

Specifies the scan link type. **Wire** specifies a wire-scan link type. **Lockup** specifies a lock-up latch-scan link type.

-test_mode *test_mode*

Specifies the test mode of the specification. The default is *current_mode*.

DESCRIPTION

The **remove_scan_link** command removes a scan link specified through the **set_scan_link** command. Scan links connect scan cells, scan segments, and scan ports within scan chains. The tool supports scan links that are implemented as wires and scan-out, and lock-up latches.

The **scan_link_name** option must uniquely identify the scan link as a design object.

To see the scan links specified, use the **report_scan_link** command.

To review the scan link specifications, use the **preview_dft** command with the **-show** option set to **cells**.

To create scan links and add them to the current design, use the **insert_dft** command.

EXAMPLES

The following example declares a scan-out, lock-up latch named *a_lckup* and then removes it with **remove_scan_link**:

```
prompt> current_design WC66
```

```
prompt> set_scan_link a_lckup Lockup
```

```
prompt> remove_scan_link a_lckup Lockup
```

SEE ALSO

- current_design(2)
- report_scan_link(2)
- set_dont_touch(2)
- set_scan_configuration(2)
- set_scan_element(2)
- set_scan_link(2)
- set_scan_path(2)
- write(2)

remove_scan_path

Removes the scan-path specification for the current design in **set_scan_path**.

SYNTAX

```
status remove_scan_path  
-chain scan_chain_name  
[-view existing_dft | spec]  
[-test_mode test_mode]
```

Data Types

```
scan_chain_name  string  
test_mode       string
```

ARGUMENTS

-chain *scan_chain_name*

Specifies to use the scan path name provided, which must be a valid, existing scan-path name. If you do not specify a chain name, the command generates an error.

-view *existing_dft* | *spec*

Indicates the view to which the specification applies. When you specify **-view existing_dft**, the specification removal refers to the existing usage of a scan chain. This is used when working with DFT-inserted designs. For example, to inform the tool that chain *A* is being removed, use the following command:

```
prompt> remove_scan_path -chain A -view existing_dft
```

When you specify **-view spec**, the default for **-view**, the specification refers to scan chains that the tool must use during DFT insertion. For example, to instruct the tool to remove scan chain *A*, use the following command:

```
prompt> remove_scan_path -chain A -view spec
```

-test_mode *test_mode*

Specifies the mode to which the specification applies. The default mode is *Internal_scan*. Specifying **all** to make the specification apply to all modes. Any mode other than the default mode or **all** must be created prior to use.

EXAMPLES

The following example sets a scan path using **set_scan_path** and then removes the scan path using **remove_scan_path** the spec view:

```
prompt> current_design core
```

```
prompt> set_scan_path 1 -view spec
```

```
prompt> remove_scan_path -view spec -chain 1
```

SEE ALSO

```
insert_dft(2)  
report_scan_path(2)  
set_scan_path(2)
```

remove_scan_register_type

Removes existing scan-register types, previously set by **set_scan_register_type**, from specified cells or from the current design.

SYNTAX

```
status remove_scan_register_type  
[cell_or_design_list]
```

Data Types

cell_or_design_list list

ARGUMENTS

cell_or_design_list

Lists the names of cells or designs from which to remove the **-type** scan-register type specification. The default is the current design. Cells or designs on this list must already have had the **-type** specification placed on them by the **set_scan_register_type** command.

DESCRIPTION

The **remove_scan_register_type** command removes existing scan-register types set on specified cells or designs by the **set_scan_register_type** command. The register types are removed without changing the design.

If you invoke **remove_scan_register_type** without any options, the specification on the current design is removed.

To determine the current settings resulting from the previous use of the **set_scan_register_type** command, use the **report_scan_register_type** command.

EXAMPLES

The following example sets and then removes a **-type** specification on the cell *U1*:

```
prompt> set_scan_register_type -type {FD1S} U1
```

```
prompt> remove_scan_register_type U1
```

SEE ALSO

`current_design(2)`
`set_scan_register_type(2)`

remove_scan_replacement

Removes the table entries specified through the **set_scan_replacement** command.

SYNTAX

```
status remove_scan_replacement  
      [non_scan_seq_cell_list]
```

Data Types

```
non_scan_seq_cell_list  list
```

ARGUMENTS

non_scan_seq_cell_list

Specifies a list of valid cells in the library that have entries in the scan-replacement table (entries are placed in the scan-replacement table by the **set_scan_replacement** command).

DESCRIPTION

The **remove_scan_replacement** command removes the scan-replacement entries for the cells in the *non_scan_seq_cell_list* from the scan-replacement table. The scan-replacement table contains one-to-one mappings of flip-flops to their equivalent scan flip-flops from the target library that are to be used by **insert_dft** or **compile -scan** when scan-replacing cell instances.

The scan replacement table entries are added by the **set_scan_replacement** command. Both **compile -scan** and **insert_dft** use this mapping table to do scan replacement. You can see the scan replacement table entries with the **report_scan_replacement** command.

EXAMPLES

In the following example, all entries for *FD1* are removed:

```
prompt> remove_scan_replacement FD1
```

SEE ALSO

current_design(2)
set_scan_replacement(2)
remove_scan_register_type(2)
report_scan_replacement(2)
reset_design(2)
target_library(3)

remove_scan_skew_group

Removes scan skew groups defined by the **set_scan_skew_group** command.

SYNTAX

```
status remove_scan_skew_group  
      scan_skew_group_list
```

Data Types

```
scan_skew_group_list  list
```

ARGUMENTS

scan_skew_group_list

Specifies the list of scan skew groups to remove. Wildcards are not supported.

DESCRIPTION

This command removes the specified scan skew groups, which were previously defined by the **set_scan_skew_group** command.

EXAMPLES

Consider a case where three scan skew groups are initially defined:

```
prompt> report_scan_skew_group
```

```
*****  
Report : Scan Skew Group  
Design : top  
Version: J-2014.09  
Date   : Wed Jul 9 06:37:41 2014  
*****
```

```
=====  
TEST MODE: all_dft
```

VIEW : Specification

=====

Scan_skew_group

Number of user-defined scan skew groups: 3

G1

G2

G3

To remove two of these scan skew groups, use the following command:

prompt> **remove_scan_skew_group {G1 G3}**

Now, only a single scan skew group remains:

prompt> **report_scan_skew_group**

Report : Scan Skew Group

Design : top

Version: J-2014.09

Date : Wed Jul 9 06:37:41 2014

=====

TEST MODE: all_dft

VIEW : Specification

=====

Scan_skew_group

Number of user-defined scan skew groups: 3

G2

SEE ALSO

report_scan_skew_group(2)

set_scan_skew_group(2)

remove_scan_suppress_toggling

Removes the existing user specifications that were provided through the **set_scan_suppress_toggling** command in terms of a list of scan flip-flops to be gated by the **insert_dft** command.

SYNTAX

status **remove_scan_suppress_toggling**

ARGUMENTS

The **remove_scan_suppress_toggling** command has no arguments.

DESCRIPTION

The **remove_scan_suppress_toggling** command removes all of the existing specifications provided through the **set_scan_suppress_toggling** command.

EXAMPLES

The following example shows how to issue the **remove_scan_suppress_toggling** command:

```
prompt> remove_scan_suppress_toggling
```

SEE ALSO

[insert_dft\(2\)](#)
[report_scan_suppress_toggling\(2\)](#)
[set_scan_suppress_toggling\(2\)](#)

remove_scenario

Removes the specified scenarios from memory.

SYNTAX

```
status remove_scenario  
    scenario_name | -all
```

Data Types

scenario_name string

ARGUMENTS

scenario_name

Specifies the name of the scenario to be deleted. You can specify only a single scenario. Wildcard characters are not supported. The scenario can be either an active or an inactive scenario.

This argument and the **-all** option are mutually exclusive; you must specify one.

-all

Specifies that all scenarios, both active and inactive, are to be deleted.

This option and the *scenario_name* argument are mutually exclusive; you must specify one.

DESCRIPTION

This command removes the specified scenario or all scenarios from memory. The command deletes all scenario-specific constraints defined in those scenarios.

Multicorner-Multimode Support

This command uses information from both active and inactive scenarios.

EXAMPLES

The following example uses the **remove_scenario** command to delete all scenarios:

```
prompt> create_scenario MODE1
```

```
prompt> create_scenario MODE2
```

```
prompt> all_scenarios
```

```
MODE 1 MODE 2
```

```
prompt> remove_scenario -all
```

```
prompt> all_scenarios
```

SEE ALSO

[all_scenarios\(2\)](#)

[current_scenario\(2\)](#)

remove_sdc

Removes all Synopsys Design Constraints (SDC).

SYNTAX

```
status remove_sdc  
[-keep_parasitics]
```

ARGUMENTS

-keep_parasitics

Specifies that the RC network has been created and it does not need to be removed.

DESCRIPTION

This command removes all constraints as defined in the current version of Synopsys Design Constraints.

The removal of annotated parasitics is expected when using the **remove_sdc** command. If you need this information after removing SDC information, reannotate the parasitics data by rerunning the **extract_rc** or **read_parasitics** command.

Multicorner-Multimode Support

This command applies to all active scenarios.

EXAMPLES

The following command removes all SDC information from the current design:

```
prompt> remove_sdc
```

SEE ALSO

current_design(2)

```
remove_attribute(2)
remove_clock(2)
remove_ideal_latency(2)
remove_ideal_network(2)
remove_ideal_transition(2)
report_constraint(2)
reset_design(2)
reset_path(2)
set_max_area(2)
set_max_delay(2)
set_max_fanout(2)
set_max_time_borrow(2)
set_max_transition(2)
```

remove_sense

Removes sense information defined on pins or cell timing arcs.

SYNTAX

```
status remove_sense
  [-type clock]
  [-clocks clock_list]
  [-all]
  pins
```

Data Types

```
clock_list  list
pins        list
```

ARGUMENTS

-type *clock*

Specifies the removal of sense from **clock** (the default) networks.

-clocks *clock_list*

Specifies the list of clocks for which the clock-sense information is removed. This option can remove only clock-sense information that has been set by using the **set_sense** command with the **-clocks** option. It does not remove clock-sense settings from pins. By default, the clock-sense information is removed for all clocks passing through the specified pins.

-all

Removes all clock sense information in the current design.

pins

Specifies the pins from which to remove clock-sense information.

DESCRIPTION

This command removes user-specified clock-sense (unateness) information from specified pins and clocks. To set clock-sense information, use the **set_sense** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes positive unateness defined on a pin named XOR/Z with respect to clock CLK1, but does not remove the negative unateness:

```
prompt> set_sense -positive -clocks [get_clock CLK1] XOR/Z  
prompt> set_sense -negative XOR/Z  
prompt> remove_sense -clocks [get_clocks CLK1] XOR/Z
```

The following example removes all unateness information in the current design;

```
prompt> remove_sense -all
```

SEE ALSO

`set_sense(2)`

remove_target_library_subset

Removes target library subset constraints from the root design or from specified instances.

SYNTAX

```
status remove_target_library_subset  
  [-object_list cells]  
  [-top]
```

ARGUMENTS

-object_list *cells*

Specifies the cells from which to remove the target library subset. The cells must be instances of hierarchical designs, as subset restriction applies to those instances and their children.

-top

Removes the target library subset from the root design. If neither the **-top** option nor the **-object_list** option is specified, the **-top** option is used by default.

DESCRIPTION

This command removes the target library subset specified by the **set_target_library_subset** command with the **-library_list** option, or the Milkyway reference library subset specified with the **-milkyway_reflibs** option from the given instances or the root design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the target library subset from the root design and from block *u1*:

```
prompt> remove_target_library_subset -object_list [get_cells u1] -top
```

The following example removes the clock_path target library subset from the root design:

```
prompt> remove_target_library_subset -clock_path -top
```

The following example removes the clock_path target library subset from block *u1*:

```
prompt> remove_target_library_subset -clock_path -object_list [get_cells u1]
```

The following example removes the clock_path target library subset from the root design and from block *u1*:

```
prompt> remove_target_library_subset -clock_path -object_list [get_cells u1] -top
```

SEE ALSO

- check_target_library_subset(2)
- compile(2)
- report_target_library_subset(2)
- set_operating_conditions(2)
- set_target_library_subset(2)
- target_library(3)

remove_terminal

Removes terminals.

SYNTAX

```
status remove_terminal  
    terminals  
    [-verbose]
```

Data Types

terminals list

ARGUMENTS

terminals

Specifies the terminals to remove.

-verbose

Prints more information.

DESCRIPTION

This command removes all specified terminals. If the input collection is empty, the command returns 0 to indicate failure.

If the specified collection of terminals contains invalid terminals, the command removes them from the collection and invokes the **remove_terminal** command with the updated collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all terminals whose name matches *clock**:

```
prompt> remove_terminal clock*  
1
```

SEE ALSO

`create_terminal(2)`
`get_terminals(2)`

remove_test_mode

Removes the mode declared by the **define_test_mode** command.

SYNTAX

```
status remove_test_mode  
      test_mode_label
```

ARGUMENTS

test_mode_label

Specifies the mode to remove.

DESCRIPTION

This command removes the mode declared by the **define_test_mode** command. The command only affects the mode and model information. It does not change the implementation of the design. In particular, for existing modes, it does not remove DFT logic. It simply removes the tool's knowledge about this logic.

EXAMPLES

The following example removes the specification test mode named *my_test_mode1*:

```
prompt> remove_test_mode my_test_mode1
```

SEE ALSO

current_test_mode(2)
define_test_mode(2)

remove_test_model

Permanently removes the test model associated with a design.

SYNTAX

```
status remove_test_model  
[-design design_name]
```

ARGUMENTS

-design *design_name*

Specifies the design from which to remove the test model. If this option is not specified, the current design is used.

DESCRIPTION

This command permanently removes the test model associated with a design. If no design name is specified, the test model for the current design is removed.

EXAMPLES

The following command removes the test model for the design named *my_design*:

```
prompt> remove_test_model -design my_design
```

SEE ALSO

`list_test_models(2)`
`read_test_model(2)`

remove_test_point_element

Removes the test-point element specification for a particular test-point type and a list of pin objects for the current design.

SYNTAX

```
status remove_test_point_element  
      list_of_design_pin_objects  
      [-type test_point_type]
```

Data Types

```
list_of_design_pin_objects  list  
test_point_type           string
```

ARGUMENT

list_of_design_pin_objects

Specifies the list of design pin objects to which the **remove_test_point_element** command applies. Wildcards and collections are supported.

-type *test_point_type*

Specifies the test-point type to which the **remove_test_point_element** command applies. The following values are valid for *test_point_type*:

```
control_0  
control_1  
control_01  
control_z01  
force_0  
force_z0  
force_1  
force_z1  
force_01  
force_z01  
observe
```

DESCRIPTION

The **remove_test_point_element** command removes a user-defined test-point specification from the design for a particular test-

point type and a list of design pin objects.

EXAMPLES

The following example shows how to remove the user-defined test-point specification for the **observe** type and for the pin *U0/Q* and *U1/Q*:

```
prompt> remove_test_point_element [list U0/Q U1/Q] -type observe
```

SEE ALSO

insert_dft(2)
preview_dft(2)
report_test_point_element(2)
set_test_point_element(2)

remove_test_protocol

Removes a test protocol from memory for the current design.

SYNTAX

```
status remove_test_protocol
  [-design design_name]
  [-test_mode test_mode_name]
```

Data Types

```
design_name    string
test_mode_name string
```

ARGUMENTS

-design *design_name*

Removes the protocol for the specified design name. When this option is not specified, the command removes the protocol from the current design.

-test_mode *test_mode_name*

Removes the test protocol from the specified test mode. When this option is not specified, the current test mode is used.

DESCRIPTION

The **remove_test_protocol** command removes a test protocol. When **-test_mode** is used, the command removes the protocol for the specified mode. Without this option, the command removes the protocol for the current mode. When **-design** is used, the command removes the protocol from the specified design. Without this option, the command removes the protocol from the current design.

EXAMPLES

The following example removes the test protocol from the current design named *my_design*:

```
prompt> remove_test_protocol
Removing test protocol from current design 'my_design'...
```

SEE ALSO

`create_test_protocol(2)`

`write_test_protocol(2)`

remove_track

Removes tracks from the current design.

SYNTAX

```
status remove_track  
-all | patterns | -layer layer [-dir X | Y]  
[-verbose]
```

Data Types

patterns list
layer collection of one item

ARGUMENTS

-all

Removes all tracks in the current design.

The *patterns*, **-all**, and **-layer** arguments are mutually exclusive; you must specify one of these arguments.

patterns

Matches the track names in the current design against the specified patterns. You can specify the patterns by using the following formats:

- An asterisk (*), which indicates all tracks
- "TRACK_*", which indicates all tracks
- Track names, which are in the format TRACK_*object_id*

The *patterns*, **-all**, and **-layer** arguments are mutually exclusive; you must specify one of these arguments.

-layer *layer*

Specifies the routing layer from which to remove the track. You can specify only one layer, either by layer name, layer number, or a collection containing one layer.

The *patterns*, **-all**, and **-layer** arguments are mutually exclusive; you must specify one of these arguments.

-dir X | Y

Specifies the direction of the routing tracks to be removed. The valid values are **X** and **Y**.

By default, the direction is the routing direction of the layer specified in the physical library.

The option must be used with the **-layer** option.

-verbose

Prints additional messages.

DESCRIPTION

This command removes the specified tracks from the current design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all tracks that are inside {{2 2} {25 25}}.

```
prompt> remove_track [get_tracks -within {{2 2} {25 25}}]
1
```

The following example removes all tracks that are located on the metal2 layer.

```
prompt> remove_track [get_tracks -of_objects METAL2] -verbose
Removing track TRACK_4683
1
```

The following example removes the routing tracks from the routing layer named m3 on the floorplan.

```
prompt> remove_track -layer m3
Warning: Direction is not specified. Using the layer preferred direction.
(MWUI-125)
1
prompt> remove_track -layer m3 -dir X
1
```

SEE ALSO

create_track(2)
get_tracks(2)
report_track(2)

remove_unconnected_ports

Removes unconnected ports or pins from cells, references, and subdesigns.

SYNTAX

```
status remove_unconnected_ports  
    cell_list  
    [-blast_buses]
```

Data Types

cell_list list

ARGUMENTS

cell_list

Specifies a list of cells or instances whose unconnected ports or pins are to be removed.

-blast_buses

Indicates that if a bus has a port that is unconnected, the bus is to be deleted, the unconnected ports are to be removed, and single-bit buses are to be created for the remaining ports.

DESCRIPTION

This command removes unconnected ports or pins from cells, references, and subdesigns. The current design must be linked and uniquified before you run this command. A port is considered unconnected if it is not connected on the inside of the design. The unconnected ports or pins of each cell in **cell_list** are removed from the cell, its reference, and its subdesign simultaneously so that the current design links after you apply **remove_unconnected_ports**.

After a port is removed, any unused nets in the top design and in the design from where the port was removed are cleaned up.

Unless you specify the **-blast_buses** option, bused ports are removed only if all ports of the bus are unconnected.

Note that if your design contains multiple-instance references to subdesigns, there is a possibility of encountering link errors when using **remove_unconnected_ports**. For example, consider the situation where two designs, A and B, reference subdesign C. Executing **remove_unconnected_ports** on design A might remove ports from subdesign C, so that design B then fails to link. The same situation occurs when design A is read in, **remove_unconnected_ports** is applied, and the original version of design A is read in again. To avoid this, uniquify the blocks or ungroup one of the designs.

This command honors **dont_touch** attributes placed on cells and designs. It is unaffected by **set_boundary_optimization** and

set_compile_directives specifications.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all unconnected ports that exist in the current design:

```
prompt> remove_unconnected_ports [find -hierarchy cell """]  
Removing port 'CI' from design 'uport1_DW01_addsub_5_0'  
Removing port 'CO' from design 'uport1_DW01_addsub_5_0'  
1
```

The following example removes all unconnected ports that exist in the current design, deletes any bus that has an unconnected port, and removes the unconnected ports of the bus:

```
prompt> remove_unconnected_ports\  
-blast_buses [find -hierarchy cell """]  
Removing port 'A_4' from design 'uport1_DW01_addsub_5_0'  
Removing port 'A_3' from design 'uport1_DW01_addsub_5_0'  
Removing port 'B_4' from design 'uport1_DW01_addsub_5_0'  
Removing port 'B_3' from design 'uport1_DW01_addsub_5_0'  
Removing port 'CI' from design 'uport1_DW01_addsub_5_0'  
Removing port 'SUM_4' from design 'uport1_DW01_addsub_5_0'  
Removing port 'SUM_3' from design 'uport1_DW01_addsub_5_0'  
Removing port 'CO' from design 'uport1_DW01_addsub_5_0'  
1
```

SEE ALSO

`remove_port(2)`

remove_upf

Removes the UPF constraints from the design. This command is only supported in dc_shell.

SYNTAX

status **remove_upf**

ARGUMENTS

The **remove_upf** command has no arguments.

DESCRIPTION

The **remove_upf** command is used to clean up UPF data before a compile command is executed for the design.

EXAMPLES

The following example shows a typical use of the **remove_upf** command:

Loading the design RTL and libraries

```
prompt> load_upf upf_file_1
```

```
prompt> remove_upf
```

```
prompt> load_upf upf_file_2
```

```
prompt> compile_ultra
```

SEE ALSO

compile_ultra(2)

load_upf(2)

remove_user_attribute

Removes a user-specified attribute from a design or library object.

SYNTAX

```
list remove_user_attribute  
  object_list  
  attribute_name  
  [-bus]  
  [-quiet]
```

Data Types

```
object_list    list  
attribute_name string
```

ARGUMENTS

object_list

Specifies a list of design or library objects from which to remove the attribute.

attribute_name

Specifies the name of the attribute to remove.

-bus

Removes the attribute from the bus as a whole, and not from each individual bus member.

-quiet

Turns off the warning message that is issued if the attribute or objects are not found.

DESCRIPTION

This command removes the user-specified attribute from the specified objects. For a complete list of attributes, see the **attributes** man page.

The command returns the list of objects from which the attribute is removed. If an empty string is returned, no object was removed.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command removes the **output_not_used** attribute on the *OUT1* port:

```
prompt> remove_user_attribute [get_ports OUT1] output_not_used  
{OUT1}
```

SEE ALSO

set_attribute(2)
set_user_attribute(2)
attributes(3)

remove_user_shape

Removes objects that are user shapes.

SYNTAX

```
status remove_user_shape
[-verbose]
user_shapes
```

Data Types

user_shapes collection

ARGUMENTS

-verbose

Prints additional information. If this option is not specified, the tool suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

user_shapes

Specifies an asterisk (*) as the pattern, and all user shapes are removed.

DESCRIPTION

This command removes user shapes. It returns 1 if successful or 0 if it fails.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes user shapes.

```
prompt> remove_user_shape *
1
```

SEE ALSO

`create_user_shape(2)`
`remove_net_shape(2)`

remove_verification_priority

Removes the **verification_priority** attribute from the specified objects.

SYNTAX

```
status remove_verification_priority  
[-all]  
object_list
```

Data Types

object_list list

ARGUMENTS

-all

Removes the **verification_priority** attribute from all of the designs and instances in the designs.

object_list

Specifies a list of objects from which the attribute is to be removed.

DESCRIPTION

This command removes the **verification_priority** attribute from the specified objects.

EXAMPLES

The following example removes the **verification_priority** attribute from all of the designs:

```
prompt> remove_verification_priority -all
```

SEE ALSO

```
get_attribute(2)
list_attributes(2)
reset_design(2)
set_verification_priority(2)
```

remove_via

Removes vias from the current design.

SYNTAX

```
status remove_via
  [-verbose]
  [vias]
```

Data Types

vias collection

ARGUMENTS

-verbose

Prints additional messages.

vias

Specifies the vias to remove. If you do not specify this argument, all vias are removed.

DESCRIPTION

This command removes the specified vias from the current design. If you do not specify any vias, the command removes all vias from the design.

This command returns 1 if successful or 0 if it fails.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **remove_via** command to remove all the vias in the current design:

```
prompt> remove_via  
1
```

SEE ALSO

- `create_net_shape(2)`
- `create_via(2)`
- `remove_net_shape(2)`

remove_via_ladder_constraints

Removes the specified via ladder constraints.

SYNTAX

```
status remove_via_ladder_constraints  
-pins collection_of_pins  
-all
```

ARGUMENTS

DESCRIPTION

This command removes via ladder lists that were previously attached to logical pins by the **set_via_ladder_constraints** command.

This command removes via ladder constraints for all the pins if **-all** is specified or for specified pins with **-pins** option. Options **-all** and **-pins** are mutually exclusive. At least one of these must be provided.

A via ladder is a physical structure of connected vias and wires that is placed on top of a standard cell pin as a special connecting device. A via ladder can have, for example, two rows of via1, four columns of via2, and one row of via3 connected with a metal4 wire.

A named via ladder is defined by set of rules in the technology file.

This command is part of the user interface for via ladder insertions.

Note that via ladder insertion is supported only by the classic router.

EXAMPLES

The following command shows removing via ladder constraints on the U1/Z pin

```
prompt> remove_via_ladder_constraints -pins U1/Z
```

SEE ALSO

set_via_ladder_constraints(2)
report_via_ladder_constraints(2)

remove_via_ladder_rules

Removes via ladders rules for the block.

SYNTAX

status **remove_via_ladder_rules**

DESCRIPTION

This command removes via ladder rules for the block set using `set_via_ladder_rules`.

This command returns 1 if succeeded, 0 otherwise.

EXAMPLES

```
prompt> remove_via_ladder_rules  
1
```

SEE ALSO

`set_via_ladder_rules(2)`
`report_via_ladder_rules(2)`

remove_via_rules

Removes via_rule objects from the current library.

SYNTAX

```
collection remove_via_rules  
[-all]  
via_rule_list
```

ARGUMENTS

-all

Removes all via rules.

RETURN VALUE

This command returns the removed via rule's count, an empty string if it fails, or a TCL_ERROR if there is a command syntax error.

DESCRIPTION

This command removes a list of via_rules from the current library.

EXAMPLES

The following example removes via rule named *VR1*

```
prompt> remove_via_rules VR1  
1
```

SEE ALSO

`create_via_rule(2)`
`get_via_rules(2)`
`report_via_rules(2)`

remove_voltage_area

Removes voltage areas from the current design.

SYNTAX

```
status remove_voltage_area  
  [-name list]  
  -all | patterns
```

Data Types

patterns list

ARGUMENTS

-name *list*

Specifies the names of voltage areas to be removed.

-all

Removes all voltage areas in the current design.

patterns

Specifies the list of hierarchical cells that represent a voltage area.

DESCRIPTION

This command removes voltage areas from the design.

When a voltage area is removed, all its hierarchical cells are also disassociated with the voltage area. After one hierarchical cell is disassociated, its leaf child cells will be reassociated with its nearest ascent's voltage area, if any. If none of its ascents is associated with any voltage area, or it does not have any ascents, its leaf child cells will be free.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes voltage areas whose names begin with *tap*:

```
prompt> remove_voltage_area tap*
```

SEE ALSO

`create_voltage_area(2)`
`report_voltage_area(2)`

remove_wire_load_min_block_size

Removes the **wire_load_min_block_size** attribute from the current design.

SYNTAX

status **remove_wire_load_min_block_size**

ARGUMENTS

The **remove_wire_load_min_block_size** command has no arguments.

DESCRIPTION

This command removes the **wire_load_min_block_size** attribute from the current design. To set the wire-load minimum block size, use the **set_wire_load_min_block_size** command, with the wire-load mode enclosed in single quotes.

Use the **remove_wire_load_model** command to remove **wire_load_min_block_size** attributes.

EXAMPLES

The following example removes the **wire_load_min_block_size** attribute from the design named *TOP*:

```
prompt> current_design TOP
```

```
prompt> remove_wire_load_min_block_size
```

SEE ALSO

current_design(2)
remove_wire_load_model(2)
remove_wire_load_selection_group(2)
reset_design(2)
set_load(2)

```
set_local_link_library(2)
set_wire_load_min_block_size(2)
auto_wire_load_selection(3)
link_library(3)
```

remove_wire_load_model

Removes wire-load model attributes from designs, ports, and hierarchical cells. This command is not supported in Design Compiler in topographical mode.

SYNTAX

```
status remove_wire_load_model
      [-min]
      [-max]
      [object_list]
```

Data Types

object_list list

ARGUMENTS

-min

Removes the wire-load model or selection group for minimum delay analysis only. If **-max** is specified, then both minimum and maximum delay wire-load models are removed.

-max

Removes the wire-load model or selection group for maximum delay analysis only. Any model set for minimum delay analysis is not affected.

object_list

Specifies a list of designs, ports, and cells from which to remove the wire-load model attributes.

DESCRIPTION

This command removes wire-load model attributes from designs, ports, and hierarchical cells of the current design (all specified in the *object_list*),

Using the **remove_wire_load_model** command is recommended, but you can also use the **reset_design** command to remove wire-load model attributes.

EXAMPLES

The following example removes wire-load model attributes from the design named *TOP*:

```
prompt> current_design TOP
```

```
prompt> remove_wire_load_model
```

The following example removes the minimum wire-load model from the design list named *DESIGN_LIST*:

```
prompt> remove_wire_load_model -min DESIGN_LIST
```

SEE ALSO

- characterize(2)
- current_design(2)
- report_lib(2)
- reset_design(2)
- set_load(2)
- set_local_link_library(2)
- set_wire_load_model(2)
- set_wire_load_mode(2)
- set_wire_load_min_block_size(2)
- set_wire_load_selection_group(2)
- auto_wire_load_selection(3)
- link_library(3)

remove_wire_load_selection_group

Removes the wire-load model selection group from designs and cells of the current design.

SYNTAX

```
status remove_wire_load_selection_group  
  [-min]  
  [-max]  
  [object_list]
```

Data Types

object_list list

ARGUMENTS

-min

Removes the selection group from designs and cells for minimum delay analysis only. Specifying neither or both **-max** and **-min** options removes both maximum and minimum wire-load selection groups.

-max

Removes the selection group from designs and cells for maximum delay analysis only. Specifying neither or both **-max** and **-min** options removes both maximum and minimum wire-load selection groups.

object_list

Removes the minimum or maximum selection group from the specified designs and cells in the list.

DESCRIPTION

This command removes the **wire_load_model_selection_group** and **wire_load_model_selection_group_lib** attributes from designs and cells. If none of designs or cells is specified, this command removes the attributes of current design.

The removal of the selection group can be for minimum delay analysis, maximum delay analysis, or both. This command removes the wire-load selection group for minimum delay and maximum delay analysis if neither **-min** nor **-max** is specified.

For a more detailed report on wire-load selection groups associated with designs or cells use the **report_wire_load** command.

EXAMPLES

The following example removes the selection group for the current design:

```
prompt> current_design TOP
```

```
prompt> remove_wire_load_selection_group 2layermetal
```

The following example removes the selection group from a subdesign named *LOW* and hierarchical cell *U1/U2* in the *TOP* design:

```
prompt> current_design TOP
```

```
prompt> remove_wire_load_selection_group {LOW U1/U2}
```

SEE ALSO

- characterize(2)
- current_design(2)
- remove_wire_load_min_block_size(2)
- remove_wire_load_model(2)
- report_lib(2)
- reset_design(2)
- set_load(2)
- set_local_link_library(2)
- set_wire_load_selection_group(2)
- link_library(3)

rename

Renames or deletes a command.

SYNTAX

```
string rename  
    old_name  
    new_name
```

ARGUMENTS

old_name

Specifies the current name of the command.

new_name

Specifies the new name of the command.

DESCRIPTION

Renames the *old_name* command so that it is now called *new_name*. If *new_name* is an empty string, then *old_name* is deleted. The *old_name* and *new_name* arguments may include namespace qualifiers (names of containing namespaces). If a command is renamed into a different namespace, future invocations of it will execute in the new namespace. The **rename** command returns an empty string as result.

Note that the **rename** command cannot be used on permanent procedures. Depending on the application, it can be used on all basic builtin commands. In some cases, the application will allow all commands to be renamed.

WARNING: **rename** can have serious consequences if not used correctly. When using **rename** on anything other than a user-defined Tcl procedure, you will be warned. The **rename** command is intended as a means to wrap other commands: that is, the command is replaced by a Tcl procedure which calls the original. Parts of the application are written as Tcl procedures, and these procedures can use any command. Commands like **puts**, **echo**, **open**, **close**, **source** and many others are often used within the application. Use **rename** with extreme care and at your own risk. Consider using **alias**, Tcl procedures, or a private namespace before using **rename**.

EXAMPLES

This example renames my_proc to my_proc2:

```
prompt> proc my_proc {} {echo "Hello"}
prompt> rename my_proc my_proc2
prompt> my_proc2
Hello
prompt> my_proc
Error: unknown command 'my_proc' (CMD-005).
```

SEE ALSO

[define_proc_attributes\(2\)](#)

rename_design

Renames a design in memory, or moves a list of designs to a file.

SYNTAX

```
status rename_design
    design_list
    [target_name]
    [-prefix prefix_name]
    [-postfix postfix_name]
    [-dont_link_with_original_name]
    [-update_links]
```

Data Types

<i>design_list</i>	list
<i>target_name</i>	string
<i>prefix_name</i>	string
<i>postfix_name</i>	string

ARGUMENTS

design_list

Specifies a list of designs to be renamed.

target_name

Specifies the new name of the design. If *design_list* has only one design, *target_name* will be the new design name. If *design_list* has multiple designs, *target_name* must be a file name and all the designs in *design_list* are moved to the specified file with their base design names. You must specify either *target_name*, or *-prefix*, or *-postfix*. These arguments are mutually exclusive; you can only specify one.

-prefix prefix_name

Specifies the prefix name that is inserted in front of design names of designs in *design_list*. One of *target_name*, *-prefix*, or *-postfix* is required, but they are mutually exclusive.

-postfix postfix_name

Specifies the postfix name that is appended to the design names of designs in *design_list*. One of *target_name*, *-prefix*, or *-postfix* is required, but they are mutually exclusive.

-dont_link_with_original_name

Specifies that the linker should not use the design's original name when it matches up reference names with design names. See

the "Interaction With the Linker" section for more information.

-update_links

This option is obsolete and will be removed in a future release. The behavior formerly enabled by this option is automatically enabled.

DESCRIPTION

The most simple use of this command is to assign a new name to a single design.

Within a file containing the design, every design name must be unique. An error occurs if the name of the new design you specified already exists in the file.

You can also use **rename_design** to move a specified list of designs to a file that can accept multiple designs.

To save these files to disk, use the **write_file** command.

Interaction With the Linker

When a design is renamed, any references to that design in the current hierarchy are updated to refer to the new design name.

By default, the design being renamed retains information about its original HDL template name and parameters. As a result, the design can continue to be instantiated by subsequent references to the old HDL template name and parameters. This behavior can be desirable if you are renaming simply to shorten a long design name, for example. If you don't want this behavior, use the **-dont_link_with_original_name** option. If you specify this option, the renamed design drops its association with the original HDL template name and parameters, and the design can only be instantiated using the new name.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example renames the design named *TEST* to *BILL*:

```
prompt> rename_design TEST BILL
Renaming design 'TEST' to 'BILL'
1
```

In this example, the **get_designs** command is used with the **rename_design** command to move all designs currently read into memory to the same file named *all.ddc*:

```
prompt> list_designs -show_file
```

Design	File	Path
-----	-----	-----
T2	T2.ddc	/usr/project
AD	AD.ddc	/test
HA	HA.ddc	/test

```
prompt> rename_design [get_designs *] /usr/project/all.ddc
Renaming design 'T2' to '/usr/project/all.ddc:T2'
Renaming design 'AD' to '/usr/project/all.ddc:AD'
Renaming design 'HA' to '/usr/project/all.ddc:HA'
1
```

```
prompt> list_designs -show_file
```

Design	File	Path
-----	-----	-----
T2	all.ddc	/usr/project
AD	all.ddc	/usr/project
HA	all.ddc	/usr/project
1		

A file must be specified as the second argument of the **rename_design** command, as shown in the following error example:

```
prompt> rename_design {A B C} all.ddc:TEST3
Error: Cannot copy multiple designs to a single design
0
```

The *-prefix* and *-postfix* options provide a convenient and efficient way to rename multiple designs. For instance, the following script prepends the string *NEW_* to the name of design *D*, and changes links for its instance cells:

```
prompt> get_cells -hierarchical -filter "ref_name == D"
{b_in_a/c_in_b/d1_in_c b_in_a/c_in_b/d2_in_c}

prompt> rename_design D -prefix NEW_
Information: Renaming design /test_dir/D.ddc:D to /test_dir/D.ddc:NEW_D. (UIMG-45)

prompt> get_cells -hierarchical -filter "ref_name == D" ;# no such cells!
prompt> get_cells -hierarchical -filter "ref_name == NEW_D"
{b_in_a/c_in_b/d1_in_c b_in_a/c_in_b/d2_in_c}
```

SEE ALSO

change_link(2)
copy_design(2)
remove_design(2)

rename_mw_lib

Renames a Milkyway library.

SYNTAX

```
status rename_mw_lib  
-from lib_name  
-to lib_name
```

Data Types

lib_name string

ARGUMENTS

-from *lib_name*

Specifies the name of the Milkyway library that is to be renamed. The specified Milkyway library must not be open in the current session.

-to *lib_name*

Specifies the new name of the Milkyway library.

DESCRIPTION

This command renames a Milkyway library.

If this library is a reference library of another main library, you may need to use the **set_mw_lib_reference** command to update reference library information in the main library.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example changes the name of the library from *access05* to *access06*:

```
rename_mw_lib
```

```
prompt> rename_mw_lib -from access05 -to access06
```

SEE ALSO

copy_mw_lib(2)

replace_clock_gates

Replaces manually-inserted clock gates with Power Compiler clock gates.

SYNTAX

```
status replace_clock_gates  
    [-global]  
    [-no_hier]
```

ARGUMENTS

-global

Performs hierarchical clock-gate replacement in all subdesigns as one global step.

-no_hier

Limits clock-gate replacement to the top level of the current design.

DESCRIPTION

This command performs clock-gating replacement of manually-inserted clock gates with Power Compiler clock gates. Before using this command, you can use the **set_clock_gating_style** command to specify the structure of the clock-gating circuitry to be inserted. If no style is explicitly set, a default clock-gating style is applied. See the man page for the **set_clock_gating_style** command for more details about the default clock-gating style.

Manually-inserted clock gates can introduce problems on the clock net, which can cause incorrect operation after synthesis. By replacing the manually-inserted clock gates with Power Compiler clock gates, correct operation of the synthesized circuit is ensured. Furthermore, the clock gates are properly recognized by Power Compiler. As such, the gate-level commands, such as **rewire_clock_gating** and **remove_clock_gating**, are available to optimize the clock-gated netlist. During physical synthesis, the clock gates are grouped together with their driven modules and registers.

Replacement of manually-inserted clock gates with Power Compiler clock gates is especially useful when you want to clock gate an entire module. In many cases such module-level clock gates cannot be inserted automatically. The **replace_clock_gates** command is recommended for creating module-level clock gating.

Before clock-gating replacement can be performed, the clock ports must be identified with the **create_clock** command. All combinational cells that are not buffers or inverters will be considered for clock gating unless they are excluded with the **set_module_clock_gates** command. Clock-gate replacement can only be applied to combinational cells that perform simple AND/NOR or OR/NAND functionality on the clock net. Also, the clock must drive registers or modules that are triggered by the same clock edge. The clock-gating style must be compatible with the functionality of the combinational cell and the edge type of the gated registers.

Only options of clock-gating style that have to do with structure of the clock gates are honored. Options that have to do with fanout or multi-stage clock-gating are not taken into account. Also, observation circuitry will not be created.

For example, with a latch-based clock-gating style, only cells with AND/NOR functionality driving positive-edge registers, and cells with OR/NAND functionality driving negative-edge registers will be replaced. If a clock net drives both positive- and negative-edge registers, or if the clock net is fed to an output port, the manually-inserted clock gates for that clock net will not be replaced. Black-box modules and modules with clock ports that drive special sequential cells (such as memory cells) can be module-level clock gated by specifying the edge type with the **set_replace_clock_gates** command. Furthermore, if a latch-free clock-gating style is selected, the enable inputs to the manually-inserted clock gate should be driven by registers that are clocked by a clock signal that is compatible with the clock input to the clock gate.

When the **-no_hier** option is omitted, clock-gate replacement is performed on all subdesigns. Every subdesign is processed independently. Otherwise, only the top level of the current design is processed.

To control the names of modules, cells, and gated clock nets created by this command, you could set different naming style variables listed at the end.

EXAMPLES

The commands in the following example show the typical flow for using the **replace_clock_gates** command. After reading or elaborating the design and defining the clock ports, you set the clock-gating style to specify how to perform clock gate replacement. The **replace_clock_gates** command replaces the manually-inserted clock gates.

```
prompt> read_verilog design.v

prompt> current_design top

prompt> link

prompt> set_clock_gating_style -sequential latch

prompt> create_clock clk

prompt> replace_clock_gates
```

SEE ALSO

- compile(2)
- create_clock(2)
- set_clock_gating_style(2)
- power_cg_cell_naming_style(3)
- power_cg_gated_clock_net_naming_style(3)
- power_cg_module_naming_style(3)

replace_synthetic

Implements all synthetic library parts of a design using generic logic.

SYNTAX

status **replace_synthetic**
[-ungroup]

ARGUMENTS

-ungroup

Ungroups all implemented synthetic library parts into their containing designs. If **-ungroup** is not specified, synthetic library parts are implemented as a level of hierarchy.

DESCRIPTION

This command will be obsolete in a future release. Use of this command is not recommended. This command processes all synthetic library parts in a hierarchical design. The processing is a subset of the synthetic library part processing performed by the **compile** command. A simple area-based resource sharing step is called on the design, followed by implementing all parts with minimum area implementations. To disable resource sharing, set the **hdlin_resource_allocation** variable to **none**. Alternatively, you can call **set_resource_allocation none** for the design.

Several high-level optimizations during compile (including timing-driven resource sharing) depend on synthetic library parts. Using **replace_synthetic** before **compile** disables these optimizations.

Use **replace_synthetic** to interface to tools that do not recognize unimplemented, synthetic library parts. Use **replace_synthetic** before using FSM **extract**, or before writing a pre-compile description for simulation.

The **replace_synthetic** command does not process references that are marked **dont_touch**.

EXAMPLES

The following example implements synthetic parts in the current design:

```
prompt> replace_synthetic
```

SEE ALSO

compile(2)
hlo_resource_allocation(3)
synthetic_library(3)

report_activity

Reports switching activity.

SYNTAX

```
int report_activity  
  [-rtl]  
  [-driver]  
  [-show_zeros]  
  [-scenarios scenario_list]
```

Data Types

scenario_list list

ARGUMENTS

-rtl

Reports switching activity similar to the **report_saif -rtl_saif** command in Design Compiler. The object types listed are port, sequential cell ("seq-cell") and tristate cells ("tri-cell").

-driver

Reports drivers based on their logic function. Two special situations occur: 1) a driver with an unknown function ("no-func") and 2) signals that have no driver at all ("no-driver"). The logic function listed are: primary input ports ("primary-input"), primary inout ports ("primary-inout"), sequential pins ("seq-pin"), tristate pins ("tri-pin"), combinational pins ("comb-pin"), unknown function ("no-func"), and no drivers ("no-driver"). If user essential activity points have been created, these will be printed in a second report as well.

-show_zeros

Include zero-valued columns in the report. By default, zero-valued columns are suppressed in the report.

-scenarios *scenario_list*

Specifies the scenarios to be considered. If this option is not specified, then only those scenarios are considered which contain the modes and corners specified through **-modes** and **-corners** options. Output of the command is generated separately for each scenario. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the command reports for the current scenario in the design.

DESCRIPTION

Reports the switching activity type for objects. The objects that are reported depends on the command options used. By default, this command mimics the **report_saif** command in Design Compiler. With the **-rtl** option this command mimics the **report_saif -rtl_saif** command in Design Compiler. With the **-driver** option this command prints out a driver-centric report.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify different scenarios, use the **-scenarios** option.

EXAMPLES

The following examples report switching activities for the current design.

```
prompt> report_activity
report_activity
Information: Updating design information... (UID-85)

*****

Report : activity
Design : counter
Version: P-2019.03-SP2-VAL
Date   : Thu Jul 11 10:56:18 2019
*****

-----
Activity Type port      net      leaf-pin
-----
simulated  32(94.12%)    32(18.08%)    66(12.77%)
annotated   0(0.00%)      2(1.13%)      7(1.35%)
inferred    0(0.00%)      0(0.00%)      0(0.00%)
propagated  0(0.00%)     116(65.54%)   351(67.89%)
default     2(5.88%)      27(15.25%)    93(17.99%)
-----
Total      34(100.00%)   177(100.00%)  517(100.00%)
```

```
prompt> report_activity -rtl
*****

Report : activity
      -rtl_saif
Design : counter
Version: P-2019.03-SP2-VAL
Date   : Thu Jul 11 10:58:47 2019
*****

-----
User      Simulated   Default
Object type Annotated (%) Activity (%) Activity (%) Total
-----
Ports      0(0.00%)    32(94.12%)  2(5.88%)    34
Seq Cells  2(13.33%)   13(86.67%)  0(0.00%)    15
Tri Cells  0(0.00%)    0(0.00%)   0(0.00%)     0
-----
```


Unlisted columns (all zeros):

inferred
propagated

prompt> **report_activity -driver**

Report : activity

Design : counter

Version: P-2019.03-SP2-VAL

Date : Thu Jul 11 11:00:31 2019

	Activity Type	primary-input	seq-pin	comb-pin	Total
simulated	32(94.12%)	13(14.44%)	53(12.41%)	98	
annotated	0(0.00%)	2(2.22%)	5(1.17%)	7	
inferred	0(0.00%)	0(0.00%)	0(0.00%)	0	
propagated	0(0.00%)	30(33.33%)	321(75.18%)	351	
default	2(5.88%)	45(50.00%)	48(11.24%)	95	

Total	34(100.00%)	90(100.00%)	427(100.00%)	551

Unlisted columns (all zeros):

primary-inout
icg-pin
tri-pin
no-func
no-driver

SEE ALSO

get_switching_activity(2)
read_saif(2)
reset_switching_activity(2)
set_switching_activity(2)

report_ahfs_options

Generates a report about the automatic high-fanout synthesis options.

SYNTAX

integer **report_ahfs_options**

ARGUMENTS

The **report_ahfs_options** command has no arguments.

DESCRIPTION

This command reports all of the options set for running automatic high-fanout synthesis. See the **set_ahfs_options** command man page for the default values of the options and a description of how each option can control the automatic high-fanout synthesis flow.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the current automatic high-fanout synthesis options:

```
prompt> report_ahfs_options
```

```
Report AHFS options:
```

```
*****
```

```
AHFS options for the design:
```

```
Enable port punching: ON
```

```
Default Reference : buffer_1
```

```
Preserve Boundary Phase : OFF
```

```
No Port Punching on these hier cells : ""
```

```
Global Route: ON
```

```
1
```

SEE ALSO

`set_ahfs_options(2)`

report_annotated_check

Displays all annotated timing checks on the current design.

SYNTAX

status **report_annotated_check**
[-nosplit]

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

This command lists all annotated timing checks in the current design. The pin-to-pin timing checks reported are setup and hold.

To list annotated delays, use the **report_annotated_delay** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following is an example of an annotated report:

```
prompt> report_annotated_check
```

```
*****
```

```
Report : annotated_check
```

```
Design : counter
```

```
Version: v3.1
```

```
Date   : Tue Apr 14 19:42:25 1992
```

Cell Name	From	To	Rise	Fall	Timing Check

U1	c	d	-0.20	-0.20	hold
U1	c	d	2.20	2.20	setup

SEE ALSO

- remove_annotated_check(2)
- report_annotated_delay(2)
- reset_design(2)
- set_annotated_check(2)

report_annotated_delay

Displays delays annotated on cells and nets of the current design.

SYNTAX

```
status report_annotated_delay
  [-cell]
  [-net]
  [-nosplit]
  [-summary]
  [-min]
```

ARGUMENTS

-cell

Reports delay data annotated on cells only. By default, both cell- and net-annotated data are reported.

-net

Reports delay data annotated on nets only. By default, both cell- and net-annotated data are reported.

-nosplit

Prevents line splitting and facilitates writing tools to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-summary

Reports the total number of cell and net delay arcs and how many are annotated with delay data. An arc is considered annotated with delays only if all the four delay values (rise, fall, min, max) are present on it.

-min

Reports minimum delay data annotated on cells and nets of the current design. If you do not use this option, the command reports maximum delay data instead.

DESCRIPTION

The **report_annotated_delay** command lists all delay data annotated on cells and nets in the current design. The cell-annotated delay data consists of pin-to-pin delays. The net-annotated delay data consists of pin-to-pin delays, net capacitance values, and net

resistance values.

Note that the delay values reported are the resulting cell and net delays used in the **report_timing** command and might be different from your annotated values if the delays were annotated using the **-load_delay net** option of the **read_sdf** or **set_annotated_delay** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following is an example of an annotated delay report:

```
prompt> report_annotated_delay
```

```
*****
Report : annotated -cell
Design : counter
Version: v3.0
Date   : Tue Apr 14 19:42:25 1992
*****
```

Cell Name	From	To	Rise	Fall
CO	A	Z	100.00	100.00

```
*****
Report : annotated -net
Design : counter
Version: v3.0
Date   : Tue Apr 14 19:42:25 1992
*****
```

Net Name	From	To	Rise	Fall	Load	Res.
h	ffc/QN	w/A	200.00	200.00	50.00	200.00
h	ffc/QN	m/B	200.00	200.00	50.00	200.00
h	ffc/QN	r/B		50.00	200.00	
m	m/Z	CO/A	200.00	200.00	40.00	100.00

SEE ALSO

read_sdf(2)
 remove_annotated_delay(2)
 reset_design(2)
 set_annotated_delay(2)

report_annotated_transition

Displays annotated transitions on all pins of the current design.

SYNTAX

```
status report_annotated_transition
-nosplit
```

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_annotated_transition** command lists annotated transition times at every pin in the current design.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following is an example of an annotated transition time report:

```
prompt> report_annotated_transition
```

```
report_annotated_transition
```

Pin Name	max_rise	max_fall	min_rise	min_fall
U0/A	10.00	-	-	17.00
U1/A	-	-	-	9.10
U1/Z	-	-	-	7.00

1

SEE ALSO

remove_annotated_transition(2)
reset_design(2)
set_annotated_transition(2)

report_app_var

Shows the application variables.

SYNTAX

```
string report_app_var  
  [-verbose]  
  [-only_changed_vars]  
  [pattern]
```

Data Types

pattern string

ARGUMENTS

-verbose

Shows detailed information.

-only_changed_vars

Reports only changed variables.

pattern

Reports on variables matching the pattern. The default is "".

DESCRIPTION

The **report_app_var** command prints information about application variables matching the supplied pattern. By default, all descriptive information for the variable is printed, except for the help text.

If no variables match the pattern, an error is returned. Otherwise, this command returns the empty string.

If the **-verbose** option is used, then the command also prints the help text for the variable. This text is printed after the variable name and all lines of the help text are prefixed with " #".

The Constraints column can take the following forms:

```
{val1 ...}
```

The valid values must belong to the displayed list.

`val \<= a`

The value must be less than or equal to "a".

`val \>= b`

The value must be greater than or equal to "b".

`b \<= val \<= a`

The value must be greater than or equal to "b", and less than or equal to "a".

EXAMPLES

The following are examples of the **report_app_var** command:

```
prompt> report_app_var sh*
Variable      Value  Type  Default  Constraints
-----
sh_continue_on_error  false  bool  false
sh_script_stop_severity none    string none    {none W E}
\\.\\.
```

```
prompt> report_app_var sh* -verbose
Variable      Value  Type  Default  Constraints
-----
sh_continue_on_error  false  bool  false
# Allows source to continue after an error
sh_script_stop_severity none    string none    {none W E}
# Indicates the error message severity level which would cause
# a script to stop executing before it completes
\\.\\.
```

SEE ALSO

get_app_var(2)
set_app_var(2)
write_app_var(2)

report_area

Displays area information for the current design or instance.

SYNTAX

```
status report_area
  [-nosplit]
  [-physical]
  [-hierarchy]
  [-designware]
```

ARGUMENTS

-nosplit

Prevents line splitting. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-physical

Reports the size of the core area and the aspect ratio of the design.

-hierarchy

Reports the area used by cells across the design hierarchy. Reports the absolute value and the percentage of area consumed by each of the cells across the hierarchy. This option also reports the details of area contribution by combinational, non-combinational, and macro or black box cells. The "black boxes" column includes macro area.

-designware

Reports the area of synthetic cells. There are two types of synthetic cells:

- Datapath cells
The datapath cells are extracted complex datapath cells.
- DesignWare singleton cells
The DesignWare singleton cells are instantiated or inferred synthetic component cells.

The report shows the total synthetic cell area and the total datapath cell area. If the synthetic cells are ungrouped during compile, the report shows the estimated area of the ungrouped synthetic cells.

DESCRIPTION

The **report_area** command lists the area statistics for the current instance or the current design. The report includes combinational, non-combinational, and total area information. If you set the **current_instance** command, the report is generated for the design of that instance. Otherwise the report is generated for the current design.

The number of combinational cells, number of sequential cells, and the number of macros/black boxes only include leaf cells.

Note that the number of macros reported by the **report_area** command is based on information from the logic libraries. This number can differ from what is returned by the **all_macro_cells** command, which is based on information from the physical libraries. For the purposes of the **report_area** command, macros and leaf-level black box cells are reported together. The **report_area** macro_cell count also includes padcells and power switches. Hierarchical black-box cells are not counted because no area is associated with them.

Also note that the core area numbers reported by the **report_area** command is the bounding box of the core area rather than the defined rectilinear shape.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example generates an area report:

```
prompt> report_area
```

```
*****
Report : area
Design : top
Version: H-2013.03-SP1
Date   : Wed Apr 17 18:44:40 2013
*****
```

Library(s) Used:

```
slow (File: /remote/dtdata1/testdata/libraries/syn/slow.db)
```

```
Number of ports:      108
Number of nets:       385
Number of cells:      256
Number of combinational cells: 254
Number of sequential cells: 0
Number of macros/black boxes: 0
Number of buf/inv:    31
Number of references: 23
```

```
Combinational area:    228294.400621
Buv/Inv area:          21384.014281
Noncombinational area: 0.000000
Macro/Black Box area:  0.000000
Net Interconnect area: undefined (Wire load has zero net area)
```

```
Total cell area:      228294.400621
Total area:           undefined
1
```

The following example generates an area report for a hierarchical design using the **-hierarchy** option:

prompt> **report_area -hierarchy**

```
*****
Report : area
Design : top
Version: H-2013.03-SP1
Date   : Wed Apr 17 18:45:01 2013
*****
```

Library(s) Used:

slow (File: /remote/dtdata1/testdata/libraries/syn/slow.db)

```
Number of ports:      108
Number of nets:       385
Number of cells:      256
Number of combinational cells: 254
Number of sequential cells: 0
Number of macros/black boxes: 0
Number of buf/inv:    31
Number of references: 23
```

```
Combinational area:    228294.400621
Buv/Inv area:         21384.014281
Noncombinational area: 0.000000
Macro/Black Box area: 0.000000
Net Interconnect area: undefined (Wire load has zero net area)
```

```
Total cell area:      228294.400621
Total area:           undefined
```

Hierarchical area distribution

```
-----
Global cell area      Local cell area
-----
Hierarchical cell  Absolute  Percent Combi-  Noncombi- Black-
                  Total    Total  national  national  boxes Design
-----
top                228294.5469  100.0  12633.5693  0.0000 0.0000 top
U1                 123539.4844  54.1   3967.9995  0.0000 0.0000 test_sel_1
U1/add_x_16_1      4550.4004  2.0    4550.4004  0.0000 0.0000 test_sel_1_DW01_add_30
U1/add_x_18_1      4998.3999  2.2    4998.3999  0.0000 0.0000 test_sel_1_DW01_add_29
U1/add_x_25_1      6054.4004  2.7    6054.4004  0.0000 0.0000 test_sel_1_DW01_add_22
U1/add_x_25_2      4390.4004  1.9    4390.4004  0.0000 0.0000 test_sel_1
U1/add_x_16_1      4550.4004  2.0    4550.4004  0.0000 0.0000 test_sel_1_DW01_add_30
U1/add_x_18_1      4998.3999  2.2    4998.3999  0.0000 0.0000 test_sel_1_DW01_add_29
U1/add_x_25_1      6054.4004  2.7    6054.4004  0.0000 0.0000 test_sel_1_DW01_add_22
U1/add_x_25_2      4390.4004  1.9    4390.4004  0.0000 0.0000 test_sel_1_DW01_add_12
U1/mult_x_10_1     21939.1992  9.6    21939.1992  0.0000 0.0000 test_sel_1_DW02_mult_J1_3
U1/mult_x_11_1     22195.1973  9.7    22195.1973  0.0000 0.0000 test_sel_1_DW02_mult_J1_2
U1/mult_x_12_1     21913.5898  9.6    21913.5898  0.0000 0.0000 test_sel_1_DW02_mult_J1_1
U1/mult_x_13_1     22732.8008 10.0    22732.8008  0.0000 0.0000 test_sel_1_DW02_mult_J1_0
U1/sub_x_17_1      5216.0000  2.3    5216.0000  0.0000 0.0000 test_sel_1_DW01_sub_6
```

```

U1/sub_x_19_1      5580.8008   2.4  5580.8008   0.0000 0.0000 test_sel_1_DW01_sub_7
U2                92121.9531  40.4 3276.7986   0.0000 0.0000 test_sel_0
U2/DP_OP_15J1_64_71 40691.3789  17.8 40691.3789   0.0000 0.0000 test_sel_0_DP_OP_15J1_64_71_0
U2/DP_OP_16J1_65_71 39392.1328  17.3 39392.1328   0.0000 0.0000 test_sel_0_DP_OP_16J1_65_71_0
U2/DP_OP_17J1_66_1425 8761.5996   3.8 8761.5996   0.0000 0.0000 test_sel_0_DP_OP_17J1_66_1425_1
-----
Total                228294.6562   0.0000 0.0000
1

```

The following example uses the **-designware** option to generate an area report for a design that contains synthetic cells:

```
prompt> report_area -designware
```

```
*****
```

Report : area

Design : top

Version: H-2013.03-SP1

Date : Wed Apr 17 18:45:20 2013

```
*****
```

Library(s) Used:

slow (File: /remote/dtdata1/testdata/libraries/syn/slow.db)

```

Number of ports:      108
Number of nets:       385
Number of cells:      256
Number of combinational cells: 254
Number of sequential cells: 0
Number of macros/black boxes: 0
Number of buf/inv:    31
Number of references: 23

```

```

Combinational area:    228294.400621
Buv/Inv area:         21384.014281
Noncombinational area: 0.000000
Macro/Black Box area: 0.000000
Net Interconnect area: undefined (Wire load has zero net area)

```

```

Total cell area:      228294.400621
Total area:           undefined

```

Area of detected synthetic parts

```

-----
Module      Implem. Count  Perc. of Area cell area
-----
DP_OP_15J1_64_71  str   1  40691.3789  17.8%
DP_OP_16J1_65_71  str   1  39392.1328  17.3%
DP_OP_17J1_66_1425 str   1  8761.5996   3.8%
DW01_add       cla   4  19993.6011   8.8%
DW01_sub       cla   2  10796.8008   4.7%
DW02_mult      csa   4  88780.7871  38.9%
-----
DP_OP Subtotal:    3  88845.1113  38.9%
Total:            1  18730.1581   8.2%

```

Subtotal of datapath(DP_OP) cell area: 107575.2694 47.1% (estimated)
Total synthetic cell area: 227146.4584 99.5% (estimated)

1

SEE ALSO

report_design(2)
set_max_area(2)

report_attribute

Reports the attributes of a cell, net, pin, port, instance, or design.

SYNTAX

```
string report_attribute  
  [-cell]  
  [-design]  
  [-hierarchy]  
  [-instance]  
  [-net]  
  [-port]  
  [-pin]  
  [-reference]  
  [-nosplit]  
  [object_list]
```

Data Types

object_list list

ARGUMENTS

-cell

Reports attributes of cells in the current instance or current design. The output is sorted alphanumerically by cell name.

-design

Reports attributes of the current instance or current design.

-hierarchy

Reports attributes for objects in the hierarchy tree.

-instance

Reports attributes of instances in the current instance or current design.

-net

Reports attributes of nets in the current instance or current design. The output is sorted alphanumerically by net name.

-port

Reports attributes of ports in the current instance or current design.

-pin

Reports attributes of pins in the current instance or current design.

-reference

Reports attributes of references in the current design.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most design information is listed in fixed-width columns. If information for a field exceeds the column width, the next field begins on a new line, starting in the correct column.

object_list

Specifies a list of objects to report.

DESCRIPTION

This command reports the attributes of specified objects. An object can be a cell, net, pin, port, instance, or design.

The *object_list* argument takes precedence over all other options. For example, the following two commands are the same:

```
prompt> report_attribute "U1" -hierarchy -cell
```

```
prompt> report_attribute "U1"
```

Instance-specific attributes are reported only at the top level of the hierarchy. If an object has no attributes, no attributes are reported. Only attributes specified directly on the design object are reported. Inherited attributes are not reported.

If you set the current instance, the report is generated for the design of that instance; otherwise the report is generated for the current design.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example displays the default attribute report for the current design:

```
prompt> report_attribute
```

```
...
```

Design	Object	Type	Attribute Name	Value
test	U1	cell	boundary_optimization	true
test	U1	cell	test_dont_fault	1
test	U1	cell	dont_touch	true
test	test	design	structure	true

```

test    test    design flatten      true
test    test    design flatten_effort  1
test    cntl    net    net_capacitance  1.000000
test    d       net    net_capacitance  0.000000
test    in      net    net_capacitance  1.000000
test    U1/A    pin    probe_point      true
test    U1/A    pin    dont_touch_network true
test    U1/Z    pin    dont_touch_network true
test    cntl    port   load              7.000000
test    d       port   load              9.000000
test    in      port   max_fanout         2.000000
test    in      port   max_transition     4.000000

```

The following example reports attributes for cells in the current design:

```
prompt> report_attribute -cell
```

...

Design	Object	Type	Attribute Name	Value
test	U1	cell	boundary_optimization	true
test	U1	cell	test_dont_fault	1
test	U1	cell	dont_touch	true

The following example reports attributes for the design named test:

```
prompt> report_attribute -design
```

...

Design	Object	Type	Attribute Name	Value
test	test	design	structure	true
test	test	design	flatten	true
test	test	design	flatten_effort	1

The following example reports attributes for nets in the current design:

```
prompt> report_attribute -net
```

...

Design	Object	Type	Attribute Name	Value
test	cntl	net	net_capacitance	1.000000
test	d	net	net_capacitance	0.000000
test	in	net	net_capacitance	1.000000

The following example reports attributes for ports in the current design:

```
prompt> report_attribute -port
```

...

Design	Object	Type	Attribute Name	Value
test	cntl	port	load	7.000000

```
test      d      port  load      9.000000
test      in     port  max_fanout  2.000000
```

The following example reports attributes for pins in the current design:

```
prompt> report_attribute -pin
```

...

Design	Object	Type	Attribute Name	Value
test	U1/A	pin	probe_point	true
test	U1/A	pin	dont_touch_network	true
test	U1/Z	pin	dont_touch_network	true

The following example reports attributes for instances in the current design:

```
prompt> report_attribute -instance
```

...

Design	Object	Type	Attribute Name	Value
UPC5	H1/U9/B	pin	test_dont_fault	2
UPC5	H1/U50	cell	test_dont_fault	1

The following example reports attributes for subdesigns:

```
prompt> report_attribute -hierarchy
```

...

Design	Object	Type	Attribute Name	Value
TOP	TOP	design	default_latch_type_exact	LD3
TOP	TOP	design	flatten	true
TOP	MINUS1	net	dont_touch	true
ADDER	U1	cell	boundary_optimization	true
ADDER	ADDER	design	default_latch_type_exact	LD3
ADDER	ADDER	design	default_flip_flop_type	la1.0(Q)
ADDER	CARRY	net	dont_touch	true
ADDER	U1	cell	boundary_optimization	true
ADDER	ADDER	design	default_latch_type_exact	LD3
ADDER	ADDER	design	default_flip_flop_type	la1.0(Q)
ADDER	CARRY	net	dont_touch	true
ADDER	U3/A	pin	dont_touch_network	true
FULL_SUBTRACTOR	U2	cell	boundary_optimization	true
FULL_SUBTRACTOR	FULL_SUBTRACTOR	design	structure	true

SEE ALSO

```
define_user_attribute(2)
get_attribute(2)
remove_attribute(2)
```

- report_attribute(2)
- report_cell(2)
- report_design(2)
- report_net(2)
- report_port(2)
- report_reference(2)
- report_timing(2)
- set_attribute(2)

report_auto_ungroup

Displays information about the cell hierarchies that have been ungrouped using the **compile** command with either the **-auto_ungroup area** or the **-auto_ungroup delay** option. Note: This command will be obsolete in a future release.

SYNTAX

```
status report_auto_ungroup
  [-nosplit]
  [-full]
```

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the auto ungroup information appears in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-full

Displays the full hierarchy in the report. By default, if there is a submodule in multiple locations in a hierarchy, the submodule's components are listed only once, with an ellipsis (...) indicating the contents of a previously displayed module.

DESCRIPTION

This command displays the cell instances that are ungrouped using the **compile** command with either the **-auto_ungroup area** or the **-auto_ungroup delay** option. The auto ungroup information is stored on the current design. To produce a report of the hierarchies that are ungrouped during the compile process, set the current design to be the same as the compiled design.

EXAMPLES

The following is an example of an auto ungroup report of the current design:

```
prompt> report_auto_ungroup

*****

Report : Auto Ungrouping
Design : hier
```

Version: 2001.08
Date : Mon Oct 9 16:44:38 2000

=====			
Instance			
Name	Cell Name	Num. Insts	
=====			
H1	sub1	6	
H2	sub2	6	
=====			

1

SEE ALSO

compile(2)

report_autofix_configuration

Reports the global AutoFix specification applied to the current design.

SYNTAX

status **report_autofix_configuration**
[-type *fix_type*]

Data Types

fix_type string

ARGUMENT

-type *fix_type*

Specifies the fixing type to report. The valid *fix_type* values are:

all
clock
set
reset
xpropagation
internal_bus
external_bus
bidirectional

The default is **all**.

DESCRIPTION

The **report_autofix_configuration** command reports the global AutoFix specification applied to the current design. By default, all fixing type specifications are reported. To report only a particular fixing type, use the **-type** option.

EXAMPLES

The following is an example of the **report_autofix_configuration** command:


```
prompt> report_autofix_configuration -type all
```

```
*****  
Report : Autofix configuration  
Design : my_current_design  
Version: A-2007.12  
Date   : Fri Dec 28 05:38:07 2007  
*****
```

```
=====
```

```
TEST MODE: all_dft  
VIEW    : Specification
```

```
=====
```

Fix type:

FixBidirectional:

Fix method: Input

Fix type: Set

Fix method: Mux

Fix latches: Disable

Fix type: Reset

Fix method: Mux

Fix latches: Disable

Fix type: Clock

Control signal: clock_autofix_mode (TestMode)

TestData signal: clock_autofix_clock_s (TestData ScanMasterClock)

Fix method: Mux

Fix latches: Disable

Fix clocks used as data: Disable

SEE ALSO

reset_autofix_configuration(2)

set_autofix_configuration(2)

report_autofix_element

Reports all local AutoFix specifications applied to the current design.

SYNTAX

```
status report_autofix_element  
  [-type fix_type]  
  [list_of_design_objects]
```

Data Types

```
fix_type           string  
list_of_design_objects list
```

ARGUMENTS

-type *fix_type*

Specifies the fixing type to report. The valid *fix_type* values are:

```
all  
clock  
set  
reset  
xpropagation  
internal_bus  
external_bus  
bidirectional
```

The default is **all**.

list_of_design_objects

Restricts the report to local specifications applied to the specified design objects. Valid object types depend on the fixing type, as follows:

```
clock - cell (hierarchical and leaf)  
set - cell (hierarchical and leaf)  
reset - cell (hierarchical and leaf)  
xpropagation - cell (hierarchical and leaf)  
internal_bus - net  
external_bus - net  
bidirectional - port
```

DESCRIPTION

The **report_autofix_element** command reports the local AutoFix specifications applied to design objects in the current design. By default, all local specifications for all fixing types are reported. To report only a particular fixing type, use the **-type** option. To restrict the report to only particular design objects, specify the object list.

EXAMPLES

The following is an example of the **report_autofix_element** command:

```
prompt> report_autofix_element
```

```
*****
Report : Autofix element
Design : my_current_design
Version: A-2007.12
Date   : Fri May 28 11:00:29 2007
*****
```

```
=====
TEST MODE: all_dft
VIEW    : Specification
=====
```

```
Designs/Instances:      u1 u3
Fix type:               Set
Control signal:         TMC (TestMode)
TestData signal:        SET (Reset)
Fix method:             Gate
Fix latches:            Enable
```

```
Designs/Instances:      a0 a1
Fix type:               Internal_bus
Control signal:         TMC (TestMode)
Fix method:             Disable_all
```

```
1
```

SEE ALSO

```
reset_autofix_element(2)
set_autofix_element(2)
```

report_background_jobs

Reports all the completed and running background jobs submitted to run by the **redirect -bg** command.

SYNTAX

```
status report_background_jobs  
[-reset]
```

ARGUMENTS

-reset

Removes the completed jobs from the list of tracked jobs.

DESCRIPTION

This command reports all the jobs that were submitted to run by the **redirect** command with the **-bg** option.

All the completed and running background job are reported.

To omit the completed jobs, use the **-reset** option with the **report_background_jobs** command.

The return value of the command is the total number of incompleted jobs. If you want to check if there is any background jobs running, you could check the return value of this command. If the return value is 0, there is no background jobs running; if the return value is greater than 0, there are background jobs running.

EXAMPLES

The following example shows a **report_background_jobs** run.

```
prompt> redirect -bg -max_cores 1 -file rep.out {source rep.tcl}  
prompt> report_background_jobs
```

The following example checks if there is any background job running.

```
prompt> set numBgJobs [eval "report_background_jobs"]  
prompt> echo $numBgJobs
```

SEE ALSO

redirect(5)

report_block_abstraction

Reports information about the specified block abstraction instance.

SYNTAX

```
status report_block_abstraction
      [block_list]
```

Data Types

block_list list

ARGUMENTS

block_list

Specifies the block abstraction instances to be reported. The option value can be a collection of block abstraction instances or name patterns.

The command fails if it cannot find any of the specified block abstraction instances.

If this option is not specified, the command reports all block abstraction instances in the current design.

DESCRIPTION

The **report_block_abstraction** command reports details about the block abstractions that are linked to the current design.

The information reported for each block includes the block reference name; block instance name; block version; full path of the library from which the block was loaded; date and time of last modification of the block; cell count and compression statistics for each of the blocks, as well as the top-level design with blocks; UPF information; TLUPlus settings; parasitic data of the block; and user-defined scenario mappings between the top level and the blocks.

Multicorner-Multimode Support

This command uses information from all scenarios.

EXAMPLES

The following example reports all the block abstraction instances in the current design:

prompt> **report_block_abstraction**

Report : Block Abstraction

Design : top

Version : G-2012.06

Date : Mon May 7 14:42:45 2012

#####

Top-level details

#####

Block Instance	Reference	Orient	Location
sub/blk1	blk1	0	(23.78, 1127.96)
sub/blk2	blk2	0	(809.74, 25.88)

Top design leaf cell count statistics:

Block instance count : 2
 Top only cell count : 184148
 Top + interface logic cell count : 239527
 Top + Block cell count : 320803
 Compression percentage : 25.34

Top-level and block scenario mapping:

There is no user-specified scenario mapping available.

#####

Block reference : blk1

#####

File name: blk1.ddc

Last modified(mm/dd/yyyy hr:min:sec): 3/6/2012 0:42:46

Leaf cell count statistics:

Interface logic cell count : 19142
 Block cell count : 46219
 Compression percentage : 58.58

TLU+ files used:

Block scenario #0: timing (mapped from top-level scenario timing)
 Min TLU+ file : min.tlup
 Max TLU+ file : max.tlup
 Tech2ITF mapping file : tech.map

Block scenario #1 : leakage (mapped from top-level scenario leakage)
Min TLU+ file : min.tlup
Max TLU+ file : max.tlup
Tech2ITF mapping file : tech.map

Block reference : blk2
#####

File name: blk2.mw/CEL/blk2:1

Last modified(mm/dd/yyyy hr:min:sec): 3/5/2012 21:44:9

Leaf cell count statistics:

Interface logic cell count : 36237
Block cell count : 90436
Compression percentage : 59.93

TLU+ files used:

Block scenario #0: timing (mapped from top-level scenario timing)
Min TLU+ file : min.tlup
Max TLU+ file : max.tlup
Tech2ITF mapping file : tech.map

Block scenario #1 : leakage (mapped from top-level scenario leakage)
Min TLU+ file : min.tlup
Max TLU+ file : max.tlup
Tech2ITF mapping file : tech.map

1

SEE ALSO

create_block_abstraction(2)
set_top_implementation_options(2)
check_block_abstraction(2)

report_boundary_cell

Reports the boundary-cell configuration specified for the current design.

SYNTAX

status **report_boundary_cell**

ARGUMENTS

The **report_boundary_cell** command has no arguments.

DESCRIPTION

This command reports the boundary-cell configuration specified for ports or core cells of the current design.

The following format is used to report the boundary-cell configuration:

Boundary Cell Structures	Status
-----	-----
Class:	<class_name>
Function:	<function_type>
Ports:	<port_name>
Cells:	<core_list>
Type:	<cell_type>
Design Name:	<design_name>
Register IO Implementation:	<impl_name>
Use Dedicated Wrapper Clock:	<boolean>
Safe State:	<state>
...	

EXAMPLES

The following example reports the boundary-cell configuration for the core wrapper class on *port1* and *port2*:

```
prompt> set_boundary_cell -class core_wrapper -type WC_D1 \  
-safe_state none B-port_list port1
```

```

1

prompt> set_boundary_cell -class core_wrapper -design my_des1 \
    -safe_state 0 -function input -port_list port2 \
    -register in_place -use_dedicated TRUE
1

```

```

prompt> report_boundary_cell

```

```

*****
Report : Boundary Cell
Design : M1
Version: W-2004.12
Date  : Thu Sep 16 09:46:38 2004
*****

```

Boundary Cell Structures	Status
-----	-----
Class:	Core Wrapper
Function:	
Ports:	port1
Cells:	
Type:	WC_D1
Design Name:	
Register IO Implementation:	
Use Dedicated Wrapper Clock:	
Safe State:	none
Class:	Core Wrapper
Function:	Input
Ports:	port2
Cells:	
Type:	
Design Name:	my_des1
Register IO Implementation:	In Place
Use Dedicated Wrapper Clock:	True
Safe State:	0

```

1

```

SEE ALSO

```

insert_dft(2)
preview_dft(2)
set_boundary_cell(2)
set_bsd_configuration(2)
set_wrapper_configuration(2)

```

report_boundary_cell_io

Displays options set by the `set_boundary_cell_io` command.

SYNTAX

status **report_boundary_cell_io**

ARGUMENTS

The **report_boundary_cell_io** command has no arguments.

DESCRIPTION

The **report_boundary_cell_io** command displays options set by the **set_boundary_cell_io** command on the current design.

The command reports the boundary-cell I/O pins set by the **set_boundary_cell_io** command.

EXAMPLES

The following is an example of the **report_boundary_cell_io** command:

```
prompt> report_boundary_cell_io
```

```
*****
Report : Boundary Cell IO
Design : what_testcase
Version: Z-2007.03
Date   : Fri Dec 21 08:44:55 2006
*****
```

```
Cell Name: what_testcase_bs_core_inst/bsr_gpio_3_inst/data_bsr_inst_U1_U1_U3
Type: Boundary
OutPI: what_testcase_bs_core_inst/what_testcase_ts_core_inst/what_testcase_core_inst/U21/Z
OutPO: what_testcase_bs_core_inst/bsr_gpio_3_inst/U11/Z
```

```
Cell Name: what_testcase_bs_core_inst/bsr_gpio_2_inst/data_bsr_inst_U1_U1_U3
```

Type: Boundary

InPI: what_testcase_bs_core_inst/what_testcase_ts_core_inst/what_testcase_core_inst/U17/A

InPO: what_testcase_bs_core_inst/bsr_gpio_2_inst/U12/Z

Cell Name: what_testcase_bs_core_inst/bsr_gpio_1_inst/data_bsr_inst_U1_U1_U3

Type: Boundary

InPI: what_testcase_bs_core_inst/what_testcase_ts_core_inst/what_testcase_core_inst/U16/A

InPO: what_testcase_bs_core_inst/bsr_gpio_1_inst/U15/Z

OutPI: what_testcase_bs_core_inst/what_testcase_ts_core_inst/what_testcase_core_inst/U24/Z

OutPO: what_testcase_bs_core_inst/bsr_gpio_1_inst/U22/Z

SEE ALSO

remove_boundary_cell_io(2)

set_boundary_cell_io(2)

report_bounds

Reports bounds in the design.

SYNTAX

```
status report_bounds  
-all | bound_objects | -name name_list
```

Data Types

```
bound_objects  list  
name_list      list
```

ARGUMENTS

-all

Reports all bounds that were created with the **create_bounds** command. Bounds from an input PDEF file are not reported with this command.

This option is mutually exclusive with the *bound_objects* argument and the **-name** option, but you must specify one.

bound_objects

Specifies the bound objects. The bound objects can be specified by the **get_bounds** command.

This argument is mutually exclusive with the **-all** option and the **-name** option, but you must specify one.

-name *name_list*

Reports bounds with the specified names.

This option is mutually exclusive with the **-all** option and the *bound_objects* argument, but you must specify one.

DESCRIPTION

The **report_bounds** command generates reports on the user-specified bounds constraints in the design, as specified by the **create_bounds** command.

If **-all** is used, all bounds in the design created by **create_bounds** are reported. Bounds created by an input PDEF file are not reported.

If a list of bound names is specified, those bounds are reported. This report includes the bound name, the type of the bound, the

effort level, and the list of cells that belong to the bound.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports bounds of names *test_1* and *test_2*:

```
prompt> report_bounds -name {test_1 test_2}
```

Report bounds:

```
*****
```

Bound Name: test_1, type: high effort auto group bound

Cells: a b c d

Bound Name: test_2, type: soft moving bound {10 20 30 40}

Cells: e f

```
*****
```

SEE ALSO

`create_bounds(2)`

`get_bounds(2)`

`remove_bounds(2)`

`set_cell_location(2)`

`update_bounds(2)`

report_bsd_buffers

Displays information about BSR signal buffer chains.

SYNTAX

status **report_bsd_buffers**

ARGUMENTS

The **report_bsd_buffers** command has no arguments.

DESCRIPTION

This command displays information about BSR signal buffer chains in a boundary-scan-inserted design. Information about the pad and BSR instance names is shown, along with buffer instance name information for each BSR signal.

EXAMPLES

The example shows the report generated by the **report_bsd_buffers** command:

```
prompt> report_bsd_buffers
```

```
*****  
Report : BSD Buffers  
Design : M1  
Version: F-2011.09-SP3  
Date   : Thu Dec 22 03:40:36 2011  
*****
```

```
Pad instance names connected with BSR cell
```

```
-----  
U11  
U16  
U17  
U10  
...
```

...

BSR instance names connected with buffers

l1/M1_BSR_top_inst/M1_out3_bsr11

l1/M1_BSR_top_inst/M1_out3_bsr10

l1/M1_BSR_top_inst/M1_out2_bsr9

...

...

clock_dr buffer instance names

l1/M1_BSR_top_inst/M1_clock_dr_bsd_buf6

l1/M1_BSR_top_inst/M1_clock_dr_bsd_buf5

...

...

update_dr buffer instance names

...

SEE ALSO

set_bsd_configuration(2)

report_bsd_compliance

Displays options set by the **set_bsd_compliance** command.

SYNTAX

integer **report_bsd_compliance**

ARGUMENTS

The **report_bsd_compliance** command has no arguments.

DESCRIPTION

The **report_bsd_compliance** command displays the options set by the **set_bsd_compliance** command on the current design.

The command reports the compliance pins and the bit pattern set for them by **set_bsd_compliance**.

EXAMPLES

The example shows the compliance report generated by **report_bsd_compliance**:

```
prompt> report_bsd_compliance
```

```
*****
Report : BSD Compliance
Design : TOP
Version: Z-2007.03
Date   : Fri Sep 22 08:44:55 2006
*****
```

```
pattern-name  port-name  port-value
```

```
-----
```

```
p1
    my_bidi    0
    my_eni     1
```

SEE ALSO

current_test_mode(2)
report_dft_configuration(2)
report_scan_configuration(2)
report_scan_path(2)
set_bsd_compliance(2)
set_dft_configuration(2)
set_scan_configuration(2)
set_scan_path(2)

report_bsd_instruction

Displays options set by the **set_bsd_instruction** command.

SYNTAX

```
status report_bsd_instruction  
[-view spec | existing_dft]  
[-instruction list_of_instruction_names]
```

Data Types

list_of_instruction_names list

ARGUMENTS

-view spec | existing_dft

Specifies the view on which to report. Valid options are **spec** and **existing_dft**. If not specified, the tool reports both the **spec** and **existing_dft** views.

-instruction *list_of_instruction_names*

Specifies the name of an instruction or a list of instructions on which to report. If no instructions are specified, the tool reports all defined instructions.

DESCRIPTION

This command displays the options set by the **set_bsd_instruction** command on the current design.

It reports on register, opcode, capture value, input and output clock conditions, signature, internal scan pin, instruction enable pins, private, and timing of the instructions.

EXAMPLES

The following command reports on both the **spec** and the **existing_dft** view:

```
prompt> report_bsd_instruction
```


The following command reports on the specified instructions in the **existing_dft** view:

```
prompt> report_bsd_instruction -view existing_dft \
      -instruction [list EXTEST_PULSE]
```

Report : BSD Instruction

Design : TOP

Version: Z-2007.03

Date : Tue Sep 12 11:33:44 2006

=====

TEST MODE: all_dft

VIEW : Existing DFT

=====

EXTEST_PULSE

Register	BOUNDARY

Opcodes	1110

Input_clock_condition	PI

Output_condition	NONE

Time	11.59

SEE ALSO

current_test_mode(2)
report_dft_configuration(2)
report_scan_configuration(2)
report_scan_path(2)
set_bsd_instruction(2)
set_dft_configuration(2)
set_scan_configuration(2)
set_scan_path(2)

report_bsd_linkage_port

Displays options set by the **set_bsd_linkage_port** command.

SYNTAX

status **report_bsd_linkage_port**

ARGUMENTS

The **report_bsd_linkage_port** command has no arguments.

DESCRIPTION

This command displays options set by the **set_bsd_linkage_port** command on the current design.

EXAMPLES

The example shows the report generated by **report_bsd_linkage_port**:

```
prompt> report_bsd_linkage_port
```

```
*****  
Report : BSD Linkage Port  
Design : TOP  
Version: Z-2007.03  
Date   : Fri Sep 22 08:44:55 2006  
*****
```

```
my_bidi  
my_ini  
my_eni  
out1
```

SEE ALSO

current_test_mode(2)
report_dft_configuration(2)
report_scan_configuration(2)
report_scan_path(2)
set_bsd_linkage_port(2)
set_dft_configuration(2)
set_scan_configuration(2)
set_scan_path(2)

report_bsd_power_up_reset

Displays options set by the **set_bsd_power_up_reset** command.

SYNTAX

status **report_bsd_power_up_reset**

DESCRIPTION

This command displays options set by the **set_bsd_power_up_reset** command on the current design. It also reports the power-up reset pins set by **set_bsd_power_up_reset**.

EXAMPLES

This example shows the report generated by the **report_bsd_power_up_reset** command:

```
prompt> report_bsd_power_up_reset
```

```
*****  
Report : BSD Power Up Reset  
Design : TOP  
Version: Z-2007.03  
Date   : Fri Sep 22 08:44:55 2006  
*****
```

```
pin-name  polarity  delay  
-----  
U4/TN    low       13
```

SEE ALSO

current_test_mode(2)
report_dft_configuration(2)
report_scan_configuration(2)


```
report_scan_path(2)
set_bsd_power_up_reset(2)
set_dft_configuration(2)
set_scan_configuration(2)
set_scan_path(2)
```

report_buffer_tree

Reports the buffer tree and its level information at the given driver pin.

SYNTAX

```
status report_buffer_tree
  [-from start_point_list | -net net_list]
  [-depth max_depth]
  [-connections]
  [-hierarchy]
  [-physical]
  [-nosplit]
```

Data Types

```
start_point_list  list
net_list          list
max_depth        int
```

ARGUMENTS

-from *start_point_list*

Specifies the starting point of the buffer tree. The starting point is defined as level 0 of the tree. The *start_point_list* must contain only pins or ports. This option and the **-net** option are mutually exclusive.

-net *net_list*

Specifies a net as the starting point of the buffer tree. The *net_list* must contain only nets. This option and the **-from** option are mutually exclusive.

-depth *max_depth*

Reports only the *max_depth* levels of buffers in the tree. The default is to report the buffer tree until a real load or hierarchical boundary is reached.

-connections

Shows the net connections in the report. By default, the net connections are not shown.

-hierarchy

Shows the buffer tree across the hierarchy. By default, the report terminates at hierarchical boundaries.

-physical

Displays the location for each pin, with the coordinates in microns.

-nosplit

Prevents line splitting, making it easier to write scripts to extract information from the report output. By default, the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_buffer_tree** command reports a buffer tree at the given driver pins or driver nets. It reports the full name of the driver pin (which includes the name of the cell), the name of the library cell, and the level of the driver relative to the starting point. The starting point is defined as level 0 of the tree.

Reporting terminates at the non-buffer pins, hierarchical boundaries (except when the **-hierarchy** option is used), or when the level reported exceeds the value specified by *max_depth*.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to report a buffer tree:

```
prompt> report_buffer_tree -from cell1/O
```

The following example shows how to report a buffer tree with net connections:

```
prompt> report_buffer_tree -connections -from cell1/O
```

SEE ALSO

- all_fanout(2)
- balance_buffer(2)
- clean_buffer_tree(2)
- create_buffer_tree(2)
- remove_buffer_tree(2)
- report_constraint(2)
- report_hierarchy(2)
- report_net_fanout(2)

report_buffer_tree_qor

Displays quality-related properties of the buffer trees at the given driver pins.

SYNTAX

```
status report_buffer_tree_qor  
[-from list_of_driving_pins_or_nets]
```

Data Types

list_of_driving_pins_or_nets list

ARGUMENTS

-from *list_of_driving_pins_or_nets*

Specifies the starting points of the buffer trees to be reported.

DESCRIPTION

The **report_buffer_tree_qor** command reports quality-related properties of the buffer trees at given driver pins or driver nets. The report is presented as a spreadsheet containing the following columns:

1. Number of sinks (non-buffer loads)
2. Number of buffers and inverters
3. Number of buffer levels
4. Buffers' area
5. Number and value of design rule violations; max_tran, max_cap and max_fanout
6. Total negative slack on the sinks
7. Worst slack among the sinks
8. Maximum immediate fanout in the tree (max. sub-tree fanout)
9. Name of the driving pin

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to report the quality of all buffer trees with the number of sinks exceeding 100:

```
prompt> report_buffer_tree_qor \
    -from [all_high_fanout -nets -threshold 100 -through]

=====
: : : ; : MT : MC : MF : : : :
# of : # of : # of: max: buf.: violation: violation: violation: : Wrst: Max : source
sinks: buf.: inv.: lvl: area: # : value: # : value: # : value: TNS : slck: sbtr: pin
=====
2023: 0: 0: 0: 0: 0: 0.00: 0: 0.00: 0: 0.00: 0.0: ** : 2023: SysClk
2023: 0: 0: 0: 0: 0: 0.00: 0: 0.00: 0: 0.00: 0.0: ** : 2023: SliceReset_
136: 9: 9: 3: 73: 0: 0.00: 0: 0.00: 0: 0.00: 10.0:-0.12: 20: Egress/
      EPortShiftReg/
      U464/Z
744: 53: 2: 4: 418: 0: 0.00: 0: 0.00: 0: 0.00: 177.1:-0.49: 20: EnMgr/
      egressShiftEn
-----
4926: 62: 11: 4: 490: 0: 0.00: 0: 0.00: 0: 0.00: 187.1:-0.49: 2023: =TOTAL=
=====
```

SEE ALSO

- all_high_fanout(2)
- create_buffer_tree(2)
- remove_buffer_tree(2)
- report_constraint(2)
- report_net_fanout(2)

report_bus

Lists the bused ports and nets in the current instance or in the current design.

SYNTAX

```
status report_bus  
[-nosplit]
```

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the output report. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

This command displays information about buses in the current instance or the current design. If the current instance has been set, the report is generated for the design of that instance. Otherwise, the report is generated for the current design.

Buses are indexed in an ascending order in the report. If a bus is named in a descending order, the numbers in the "Step" column are negative. Bused pins are not listed in the report because the bus commands operate only on design ports (not reference ports).

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command displays bus information on the current design:

```
prompt> report_bus
```

```
*****
```

```
Report : bus
```

```
Design : SUBTRACTOR
```

```
Version: v2.0
Date  : Fri Mar 16 14:22:01 1991
*****
Bussted Port  Dir    From  To  Step Width Type
-----
A           in      0    3    1    4 array
B           in      0    3    1    4 array
DIFF        out      0    3    1    4 array

Bussted Net   From  To  Step Width Type
-----
A           0    3    1    4 array
B           0    3    1    4 array
DIFF        0    3    1    4 array
```

SEE ALSO

```
create_bus(2)
remove_bus(2)
```

report_cache

Reports on the contents of synthetic library caches.

SYNTAX

```
status report_cache
  [-design_lib list]
  [-module list]
  [-implementation list]
  [-parameters parameter_list]
  [-tech_lib list]
  [-wire_load list]
  [-operating_conditions list]
  [-directory dir_list]
  [-smaller size | -larger size]
  [-accessed_since days
   | -accessed_beyond days]
  [-netlist_only | -model_only]
  [-sort_largest | -sort_oldest | -sort_cache_key]
  [-statistics]
```

Data Types

<i>list</i>	<i>dir_list</i>
<i>parameter_list</i>	<i>list</i>
<i>dir_list</i>	<i>list</i>
<i>size</i>	<i>integer</i>
<i>days</i>	<i>float</i>

ARGUMENTS

-design_lib *list*

Restricts the report to the design libraries in the list. Strings in the list can contain the * (asterisk) wildcard character; for example, { DW_01 DW_ *SNPS* }.

-module *list*

Restricts the report to the modules in the list. Strings in the list can contain the * (asterisk) wildcard character.

-implementation *list*

Restricts the report to the implementations in the list. Strings in the list can contain the * (asterisk) wildcard character.

-parameters *parameter_list*

Restricts the report to the parameters in the list.

-tech_lib *list*

Restricts the report to the technology libraries in the list. Strings in the list can contain the * (asterisk) wildcard character.

-wire_load *list*

Restricts the report to the wire loads in the list. Strings in the list can contain the * (asterisk) wildcard character.

-operating_conditions *list*

Restricts the report to the operating conditions in the list. Strings in the list can contain the * (asterisk) wildcard character.

-directory *dir_list*

Uses the specified list of directories, instead of those listed by the **cache_read** and **cache_write** variables, as the cache directories on which to report.

-smaller *size*

Restricts the report to cache elements that are smaller than the specified number of bytes.

-larger *size*

Restricts the report to cache elements that are larger than the specified number of bytes.

-accessed_since *days*

Restricts the report to cache elements whose last access time is between now and the specified number of days ago.

-accessed_beyond *days*

Restricts the report to cache elements whose last access time is more than the specified number of days ago.

-netlist_only

Reports only on netlists.

-model_only

Reports only on models.

-sort_largest

Sorts the report by cache element size.

-sort_oldest

Sorts the report by cache element access time.

-sort_cache_key

Sorts the report by cache key. The cache key consists of the following fields:

- Root UNIX directory
- Design library
- Module,
- Implementation
- Parameters
- Technology library
- Wire load

- Operating conditions

The root directory and parameters are not used in the sorting.

-statistics

Reports timing and area information of the cached models.

DESCRIPTION

The **report_cache** command reports on the cache elements in the directories listed by the **cache_read** and **cache_write** variables. If the directory named *dir* is listed as a cache directory, the directory *dir/synopsys_cache_v*. ** is the top of the cache directory structure.

The two types of elements stored in the cache are netlists and models. Netlists are characterized by not being mapped to technologies and having no wire load or operating conditions. Models, however, always have wire load and operating condition settings. They contain timing models and areas of the corresponding netlists. For models, wire load and operating conditions can have a value of *default*. For wire load, *default* means wire-load selection was used if it is defined in the technology library or no wire load was used. For operating conditions, *default* means that *nom_temperature*, *nom_process*, and *nom_voltage* was used.

Without command arguments, all cache elements in the cache directories are listed. If there are command arguments that restrict the report, the cache element must pass each command argument specified. An element is accepted by a command argument if it matches one of the items in the command argument's list. If the **-statistics** option is specified, only cached models are listed.

For example, the following command lists the ripple adder and the Wallace Tree multiplier:

```
report_cache -module {DW01_add DW02_mult} -implementation {wall rpl}
```

The **-parameters** option has additional features. The *parameter_list* is a space-separated list of quoted parameter specifications (for example, {N=8M=6 N=3}). Each quoted parameter specification is a comma-separated list of parameter settings. Parameter settings are a parameter name followed by an equal sign followed by a parameter value. Parameter settings can also support ranges for the parameter value such as "N = [8;17]" where "[" means the endpoint is inclusive and "(" means the endpoint is exclusive. Thus, if you execute **report_cache -parameters {"N = [8;17]"}**, cache elements whose parameter *N* is greater than or equal to 8 and less than 17 are matched. The braces ({ }) are important. Since '-parameters "n=8, m=6"' get parsed as '-parameters { "n=8" "m=6" }' instead of '-parameters { "n=8, m=6" }', you cannot match a cache element with 2 parameters, *n* and *m*, which are the braces in this case. The special parameter specification "no_parameters" also matches elements with no parameters. For example, {"n=8, no_parameters"} matches elements with no parameters or matches elements that have the parameter "n=8".

Matching parameter names has additional complications because cache elements have a variable number of parameters and the parameter names vary. For example, an adder has one parameter, *n*, whereas a multiplier has two parameters, *n* and *m*. Therefore, parameter specifications are matched using two rules. First, if the names and values of a parameter specification match the cache element parameters exactly, the element is accepted. Second, if all cache element parameters are covered in a parameter specification but there are extra parameters in the parameter specification, the cache element is also matched. Otherwise, the cache element is not accepted. Thus, in the previous example, you can match both the adder and multiplier by using a parameter specification that includes both the parameters *n* and *m*. Omitting the **-parameters** option causes all parameter specifications to be accepted.

EXAMPLES

The default output format lists cache elements clustered by directory, technology library, wire load, and operating conditions. The following is an example of the default output:

```
prompt> report_cache -larger 5000
```

```
*****
Report : cache
Version: v3.1
Date   : Wed Mar 31 14:17:03 1993
*****
```

```
=====
| DESIGN | MODULE   | IMPLEMENT- | PARAMETERS | ACCESS | SIZE |
| LIBRARY |          | ATION      |            | days | bytes |
=====
```

```
Cache Directory Root: '/remote/rd30/df/tcache/'
```

```
Technology Library: 'and_or.db'
```

```
Wire Load: no_wire_load
```

```
Operating Conditions: no_operating_conditions
```

```
| DW01 | DW01_ADD_ABC | str |      | 0.0 | 5203 |
| DW01 | DW01_absval | cla | width=3 | 0.0 | 10064 |
| DW01 | DW01_add | cla | width=5 | 0.0 | 17376 |
| DW01 | DW01_inc | cla | width=3 | 0.0 | 9136 |
| DW01 | DW01_inc | cla | width=6 | 0.0 | 10912 |
| DW02 | DW02_mult | csa | A_width=3 | 0.0 | 29440 |
|      |      |      | B_width=3 |      |      |
```

```
=====
Cache Directory Root: '/remote/rd30/df/tcache/'
```

```
Technology Library: 'and_or.db'
```

```
Wire Load: '-default-'
```

```
Operating Conditions: '-default-'
```

```
| DW01 | DW01_ADD_ABC | str |      | 0.0 | 6162 |
| DW01 | DW01_GP_SUM | str |      | 0.0 | 5302 |
| DW01 | DW01_MUX | str |      | 0.0 | 5134 |
| DW01 | DW01_XOR2 | str |      | 0.0 | 5020 |
| DW01 | DW01_absval | cla | width=3 | 0.0 | 9768 |
| DW01 | DW01_add | cla | width=5 | 0.0 | 20982 |
| DW01 | DW01_inc | cla | width=3 | 0.0 | 10469 |
| DW01 | DW01_inc | cla | width=6 | 0.0 | 13654 |
| DW02 | DW02_mult | csa | A_width=3 | 0.0 | 31452 |
|      |      |      | B_width=3 |      |      |
```

```
1
```

Queries can be substantially more complicated, as in the following example:

```
prompt> report_cache -dir ~ -design_lib DW02 \
  -mod *mult -accessed_since 21 \
  -param {"n=[3;100], m=[9;100]}" "n=(6; 100), m= (6; 100)"}

```

For more information, see the DesignWare and the Library Compiler documentation.

SEE ALSO

create_cache(2)

```
remove_cache(2)
report_synlib(2)
set_wire_load_min_block_size(2)
set_wire_load_mode(2)
set_wire_load_model(2)
set_wire_load_selection_group(2)
cache_read(3)
cache_write(3)
```

report_case_analysis

Reports case analysis on ports or pins.

SYNTAX

```
string report_case_analysis  
  [-all]  
  [-nosplit]
```

ARGUMENTS

-all

Reports the built-in constant pins of the design that are considered for startpoints of constant propagation. Propagation of logic constants is only done if the **set_case_analysis** command was executed, or the variable **case_analysis_with_logic_constants** or **disable_case_analysis** is set to **true**.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_case_analysis** command reports case analysis entries specified with the **set_case_analysis** command. The report lists all pins and ports on which case analysis is specified. The list does not report where the logic constant values are propagated.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example specifies that pins *U1/U2/A* and *U1/U3/C1* are set to a constant of logic 1:

```
prompt> set_case_analysis 1 {U1/U2/A U1/U3/C1}
```

prompt> **report_case_analysis**

Report : case_analysis
Design : middle
Version: 1997.01
Date : Mon Jul 22 17:04:17 1996

Pin name	User case analysis value

U1/U2/A	1
U1/U3/CI	1

SEE ALSO

remove_case_analysis(2)
set_case_analysis(2)

report_cell

Displays information about cells in the current instance or current design.

SYNTAX

```
status report_cell
  [-nosplit]
  [-connections]
  [-verbose]
  [-physical]
  [-only_physical]
  [-significant_digits digits]
  [cells]
```

Data Types

<i>digits</i>	integer
<i>cells</i>	collection

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-connections

Displays all pins of the cell and the connected nets.

-verbose

Displays verbose connection information. For each input pin, the tool displays the net and the driver pins on that net. For each output pin, the tool displays the net and the load pins on that net.

-physical

Reports the orientation and location of the cell.

-only_physical

Reports information for physical-only cells.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are reported. Allowed values are 0 through 13 and the default is 6. Using this option overrides the value set by the **report_default_significant_digits** variable.

cells

Specifies the cells and instances to show in the report. If not specified, all cells in the current instance are listed.

DESCRIPTION

This command displays information and statistics about cells in the current instance or current design. If *current_instance* is set, the report is generated for the design of that instance; otherwise the report is generated for the current design.

Parameterized cells are attributed by the letter *p* and contain parameters that control aspects of the designs they instantiate. The actual values of a cell's parameters appear at the bottom of the report.

Control logic cells are designated as type *c* in the report generated by **report_cell**. Use this information to identify control logic cells when using the **group** command. Group control logic cells with the cells that they control.

This command does not display constant cells. To display constant cells, use the following command:

```
get_cells -hierarchical * \
-filter "@ref_name==**logic_0** || @ref_name==**logic_1**"
```

Use the **-connections** option to list the nets connected to each pin.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following is an example of a cell report:

```
prompt> report_cell
```

```
*****
Report : cell
Design : CONTROL
Version: E-2010.12
Date   : Wed Sep 22 14:09:15 2010
*****
```

Attributes:

```
  b - black box (unknown)
  bo - allows boundary optimization
  BO - reference allows boundary optimization
  c - control logic
  d - dont_touch
  D - reference or design is dont_touched
  h - hierarchical
  n - noncombinational
```


p - parameterized
 r - removable
 S - synthetic module
 u - contains unmapped logic

Cell	Reference	Library	Area	Attributes

IDIN0	NAND2	tech_lib	1.00	
IDIN1	JPMP		0.00	h, u
IDIN2	JMM		0.00	b
IOU1	NAND2	tech_lib	1.00	
IOU2	NAND2	tech_lib	1.00	
IOU3	NAND2	tech_lib	1.00	
R1	NAND3	tech_lib	2.00	
R2	INV	tech_lib	1.00	
U0	MX1P	tech_lib	8.00	n
U1	MX1P	tech_lib	8.00	d, n, 1
U3	MX1P	tech_lib	8.00	d, n
U5	MX1P	tech_lib	8.00	n
U6	MX1P	tech_lib	8.00	n, 2
U7	MX1P	tech_lib	8.00	n
U9	JMM1		0.00	b
U10	JMM1		0.00	b
U15	CTLX		101.00	h
P1	DW01_add		50.00	S, p

Total 18 cells			206.00	

Flip-Flop Types:

- 1 - Default type MX1, MX1P, MX2, MX2P, MX3, MX3P, MX4, MX4P
- 2 - Exact type LMNO1

HDL Parameters:

P1 - width => 16

The following is an example of a verbose cell connection report for the t_bar cell:

prompt> **report_cell -connections -verbose {t_bar}**

Report : cell

-connections

-verbose

Design : counter

Version: E-2010.12

Date : Wed Sep 22 21:28:33 2010

Connections for cell 't_bar':

Reference: IVA

Library: lsi_10k

Area: 1

dont_touch: FALSE

Input Pins	Net	Net Driver Pins	Driver Pin Type

A	t	t/Z	Output Pin (EO)

Output Pins	Net	Net Load Pins	Load Pin Type
-----	-----	-----	-----
Z	t_bar	zero/B	Input Pin (OR2)
1			

The following command reports on all of the registers in the current design:

```
prompt> report_cell [all_registers]
```

SEE ALSO

- all_registers(2)
- current_design(2)
- current_instance(2)
- report_compile_options(2)
- report_design(2)
- report_hierarchy(2)
- report_net(2)
- report_reference(2)
- report_default_significant_digits(3)

report_cell_mode

Generates a report of the instance modes.

SYNTAX

```
status report_cell_mode
      [-nosplit]
      [instance_list]
```

Data Types

instance_list list

ARGUMENTS

-nosplit

Prevents line splitting. By default, most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

instance_list

Indicates that the mode report is to include only the specified cells. By default, all cells that have modes are included.

DESCRIPTION

This command reports which cells have modes, and for each mode, shows whether the mode is currently enabled or disabled. This reports reflects the mode specifications of the **set_cell_mode** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example reports that two cells in the design have modes: the *Uram1* cell and the *Uram2* cell.

```
prompt> report_cell_mode
```

```
*****
Report : mode
Design : top_design
*****
```

Cell	Mode(Group)	Status
Uram1/core(RAM2_core)	read(rw)	ENABLED
	write(rw)	ENABLED
Uram2/core(RAM2_core)	read(rw)	ENABLED
	write(rw)	ENABLED

The following example reports that modes are specified for the two RAMs that have modes in the design. Ram *Uram1* is set in mode read, and Ram *Uram2* is set in mode write. This means all timing arcs of RAM *Uram1* associated with mode read are enabled, and all timing arcs associated with mode write are disabled.

```
prompt> set_cell_mode read Uram1/core
```

```
prompt> set_cell_mode write Uram2/core
```

```
prompt> report_cell_mode
```

```
*****
Report : mode
Design : top_design
*****
```

Cell	Mode(Group)	Status
Uram1/core(RAM2_core)	read(rw)	ENABLED
	write(rw)	disabled
Uram2/core(RAM2_core)	read(rw)	disabled
	write(rw)	ENABLED

SEE ALSO

```
reset_cell_mode(2)
set_cell_mode(2)
```

report_check_library_options

Reports the values or the status of the options set by the **set_check_library_options** command.

SYNTAX

```
status report_check_library_options  
  [-physical]  
  [-logic_vs_physical]  
  [-logic]  
  [-default]
```

ARGUMENTS

-physical

Reports the values or the status of the options set by the **set_check_library_options** command to check between physical libraries.

-logic_vs_physical

Reports the values or the status of the options set by the **set_check_library_options** command to check between logical libraries and physical libraries.

See the **-logic_vs_physical** option in the **set_check_library_options** man page for more information.

-logic

Reports the values or the status of the options set by the **set_check_library_options** command to check between logical libraries.

See the **-logic** option in the **set_check_library_options** man page for more information.

-default

Reports the default values of all library-checking options. If this option is not specified, the current values of the options are reported.

DESCRIPTION

The **report_check_library_options** command reports the values or the status of the options set by the **set_check_library_options** command. The **report_check_library_options** command provides information about options used for checking between logical libraries, options used for checking between physical libraries, and options used for checking between logical libraries and physical

libraries.

Use the **report_check_library_options** command after running **set_check_library_options**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, **report_check_library_options** checks all data for the specified *test.mw* library, and then reports the option values:

```
prompt> set_check_library_options -cell_footprint -bus_delimiter
```

```
prompt> report_check_library_options -logic_vs_physical
1
```

```
Report options for check_library
```

```
*****
```

```
Check cell area      : false
Check cell footprint : true
Check bus delimiter  : true
Cell names           : (null)
Logic libraries      : (null)
Physical libraries   : (null)
```

```
1
```

SEE ALSO

[check_library\(2\)](#)

[set_check_library_options\(2\)](#)

report_clock

Displays clock-related information.

SYNTAX

```
status report_clock
  [-attributes]
  [-skew]
  [-nosplit]
  [-groups]
  [-scenarios scenario_list]
  [clock_names]
```

Data Types

```
scenario_list  list
clock_names   collection
```

ARGUMENTS

-attributes

Lists all clocks in the current design, and for each clock, shows the clock name, period, waveform, source, and clock attributes such as "p" for propagated clock or "G" for generated clock. This is the default report.

-skew

Reports the clock network skew information set on the design by the **set_clock_uncertainty** command. This information includes rise delay, fall delay, plus uncertainty, and minus uncertainty of the clock network.

-nosplit

Prevents line splitting. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-groups

Reports the clock groups information set on the design by the **set_clock_groups** command.

-scenarios *scenario_list*

Reports clocks for the specified scenarios of a multi-scenario design. Inactive scenarios are skipped in the report. Each scenario is reported separately. If you do not specify this option, only the current scenario is reported. The **-scenarios** option and *clock_names* argument are mutually exclusive.

clock_names

Lists the clocks that are to be reported. Substitute the collection you want for *clock_names*.

DESCRIPTION

This command displays all clock-related information for a design. The report contents are controlled by the options used. If you do not specify any option, the attributes report is displayed by default.

The **report_transitive_fanout -clock_tree** report shows the fanout network of every clock source in the design.

To view a list, or create a collection, of all clocks in the design, use the **all_clocks** command.

The **-scenarios** option and *clock_names* argument are mutually exclusive.

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following are examples of clock reports.

prompt> **report_clock -attributes**

```
*****
Report : clocks
Design : top
Version: E-2010.12
Date   : Fri Oct 1 20:19:06 2010
*****
```

Attributes:

```
d - dont_touch_network
f - fix_hold
p - propagated_clock
G - generated_clock
g - lib_generated_clock
```

Clock	Period	Waveform	Attrs	Sources
phi1	10.00	{0 5}	{phi1}	
phi2	10.00	{5 10}	{phi2}	
off_chip_clk	50.00	{0 25}	{}	

1

prompt> **report_clock -skew**

```
*****
Report : clock_skew
```


Design : top
 Version: E-2010.12
 Date : Fri Oct 1 20:20:44 2010

Object	Rise Delay	Fall Delay	Min Rise Delay	Min fall Delay	Uncertainty Plus	Minus
phi1	0.50	0.40	0.50	0.40	0.20	0.10
phi2	0.20	-	0.20	-	-	-
ff1/CP	-	-	-	-	0.10	0.15
ff2/CP	-	-	-	-	0.10	0.15
ff3/CP	-	-	-	-	0.10	0.15
ff4/CP	-	-	-	-	0.10	0.15

Object	Max Transition		Min Transition	
	Rise	Fall	Rise	Fall
clk	0.30	0.30	-	-
1				

prompt> **report_clock -groups**

 Report : clocks
 Design : test
 Version: E-2010.12
 Date : Fri Oct 1 20:29:09 2010

Attributes:

d - dont_touch_network
 f - fix_hold
 p - propagated_clock
 G - generated_clock
 g - lib_generated_clock

Clock	Period	Waveform	Attrs	Sources
clk1	4.00	{0 2}	{clk1}	
clk2	4.00	{0 2}	{clk2}	
clk3	4.00	{0 2}	{clk3}	
clk4	4.00	{0 2}	{clk4}	

Clock Groups :

Total logically exclusive groups: 1

NAME : clk4_clk2

False timing paths.

-groups {clk4}

-groups {clk2}

Total physically exclusive groups: 1

NAME : clk3_1

False timing paths.

```
-groups {clk3}
-groups {clk4}
```

Total asynchronous groups: 1

NAME : clk1_clk2

False timing paths.

```
-groups {clk1}
-groups {clk2 clk4}
```

```
1
prompt> report_clock CLK_H
*****
```

```
Report : clocks
Design : top
Scenario(s): s2
Version: G-2012.06
Date  : Wed Jun 27 02:40:21 2012
*****
```

Attributes:

```
d - dont_touch_network
f - fix_hold
p - propagated_clock
G - generated_clock
g - lib_generated_clock
```

Clock	Period	Waveform	Attrs	Sources	Scenario
CLK_H	5.00	{0 2.5}	{clk1}	s2	

```
1
```

SEE ALSO

```
all_clocks(2)
create_clock(2)
remove_clock(2)
report_constraint(2)
report_design(2)
report_transitive_fanout(2)
set_clock_uncertainty(2)
set_clock_groups(2)
```

report_clock_gating

Reports the Power Compiler tool's clock-gating details.

SYNTAX

```
status report_clock_gating
  [-no_hier]
  [-verbose]
  [-gated]
  [-ungated]
  [-gating_elements]
  [-only cell_list]
  [-nosplit]
  [-physical]
  [-multi_stage]
  [-style]
  [-structure]
  [-scenarios scenario_list]
  [-enable_conditions]
```

Data Types

```
cell_list    list
scenario_list list
```

ARGUMENTS

-no_hier

Reports the clock-gating information only for the top level of the current design. If this option is not specified, all levels of hierarchy, starting from the current design, are reported.

-verbose

Reports more detailed information such as, test information and a list of related registers to the clock-gating cells. By default, this information is not included in the report.

-gated

Reports the names of all gated registers, along with the names of the corresponding clock-gating elements.

-ungated

Reports the names of all ungated registers, along with the unmet clock-gating conditions.

-gating_elements

Reports the names of all clock-gating elements, along with their style, setup, and hold times, inputs, and outputs.

-only *cell_list*

Reports information only for the clock-gating cells or gated and ungated registers listed in *cell_list*. By default, information is displayed for all clock-gating cells and ungated and gated registers.

-nosplit

Prevents line splitting and facilitates writing applications to extract information from the report output. Most design information is listed in fixed-width columns. If information for a field exceeds the column width, the next field begins on a new line, starting in the correct column.

-physical

Reports the physical locations and distance statistics of clock-gating cells and gated registers, along with other clock-gating information. This option can be used only with the **-gating_elements** option.

-multi_stage

Reports multistage clock-gating statistics in the clock-gating summary.

-style

Reports the clock-gating style of all clock gates in the current design.

-structure

Reports the clock-gating structure summary and the clock-gating structure details. Information about the clock-gating cells and gated registers is classified according to the clock domains where they belong. Additional information regarding fanout and clock latency on each clock-gating element is also displayed.

This option can only be combined with the **-scenarios** option.

-scenarios *scenario_list*

Produces a set of reports for the specified scenarios of a multicornner-multimode design when used with the **-structure** option. To get reports for all active scenarios, use the **[all_scenarios]** with the **-scenarios** option. If you do not specify this option, only the current scenario is reported.

-enable_conditions

Reports the enable conditions of all clock-gating cells connected to registers. A Boolean equation is used to represent the enable conditions as a sum of products: '&' is the AND operation, '|' is the OR operation and, '!' is the NOT operation. This option can only be combined with the following options: the **-only** option to report specific clock gates and registers, the **-nosplit** option to display enable conditions on a single line, and the **-high_effort** option to maximize the terms to show.

DESCRIPTION

The **report_clock_gating** command reports information about clock-gating cells and gated and ungated registers in the current design. To generate the report, the **report_clock_gating** command uses clock-gating attributes added on the clock-gating cells, by Power Compiler. Any clock gating added by other means, such as clock gating inferred by or instantiated in the RTL code, lacks these attributes and does not appear in the report.

XOR self-gating cells inserted by Power Compiler are also excluded from this report. Use **report_self_gating** command to get information about them.

When you run the command without any option, the report lists the number of clock-gating elements, the number of gated and ungated registers with percentages of the total, and the total number of registers.

When you use the **-multi_stage** option, the summary also includes statistics of multistage clock-gating. These statistics include the number of multistage clock-gating elements, such as clock gates that have other clock gates in their fanout, the average multistage fanout, the number of gated cells (registers and modules), and the maximum and average number of stages per gated cell. Use one or more of the other options to get more detailed information.

When you use the **-gating_elements** option, a "(d)" flag next to the name of a clock-gating cell indicates that the clock-gating cell or its parent hierarchical cell is marked `dont_touch` with the **set_dont_touch** command. This means that this clock-gating cell is not modified or removed.

When you use the **-structure** option, the command generates the clock-gating structure summary and clock gating structure details in the reports. The clock-gating structure summary reports the total number of registers (gated and ungated) and the number of clock gates and gated elements for each clock-gating stage, for each clock domain. The clock-gating structure details shows, for each clock domain and each clock gating stage, all clock-gating cells, their respective fanout, clock latency, and gated elements. This option can be combined only with the **-scenarios** option.

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following example shows a report generated when the **report_clock_gating** command is issued without any option. The report shows that out of 6 registers, 4 are gated and 2 are ungated.

```
prompt> report_clock_gating
```

```
*****
Report : clock_gating
Design : top
Version: E-2010.12
Date  : Fri Oct 1 20:45:04 2010
*****
```

Clock Gating Summary			

	Number of Clock gating elements		1
	Number of Gated registers		4 (66.67%)
	Number of Ungated registers		2 (33.33%)
	Total number of registers		6

The following example shows a report generated with the **-multi_stage** and **-no_hier** options for a hierarchical multistage clock-gated design. A multistage clock gate is a clock-gating cell that drives another clock-gating cell. The report shows three clock-gating elements, eight gated registers, and no ungated registers at the top level. Two of the three clock gates are multistage and their average fanout is 1.0, indicating that the clock path consists of a chain of three clock gates. There is one gated module in addition to the eight gated registers. The eight registers have three stages on their clock path, but the module has only two, bringing the

average number of stages to $2.9 = ((8*3 + 2*1)/9)$.

prompt> **report_clock_gating -multi_stage -no_hier**

```
*****
Report : clock_gating
       -no_hier
       -multi_stage
Design : top_level
Version: 2003.12
Date   : Tue Oct 21 15:45:26 2003
*****
```

Information : The design is hierarchical. Omit -no_hier option for more than one level.

Clock Gating Summary			
Number of Clock gating elements	3		
Number of Gated registers	8 (100.00%)		
Number of Ungated registers	0 (0.00%)		
Total number of registers	8		
Number of multi-stage clock gates	2		
Average multi-stage fanout	1.0		
Number of gated cells	9		
Maximum number of stages	3		
Average number of stages	2.9		

The following example shows a report generated by the **report_clock_gating** command (with no options) when multi-bit registers are present in the design. The report shows the **Clock Gating Summary** table and an additional table titled **Clock Gating Multibit Decomposition**, with information about single-bit and multi-bit registers found in the design. This additional table is present only if multi-bit registers are found in the design.

There are 26 single-bit registers in the example, 3 of them are ungated. Also, there are 7 multi-bit registers in the design, two of them are 2-bit registers and the rest are 4-bit. Both 2-bit registers are gated, while only four of the 4-bit registers are gated.

The **Clock Gating Multibit Decomposition** table has a column titled **Actual Count** which represents the physical cell count. On the other hand, the **Single-bit Equivalent** column shows how the count would look if every multi-bit register were converted to the single-bit equivalent ones. Notice that the counts in the **Clock Gating Summary** table are based on the **Single-bit Equivalent** column; this is to make easier the comparison of different clock-gating schemes independently of multi-bit register availability in the library that is being used. It is the **Single-bit Equivalent** count what is taken into account for load balancing purposes.

prompt> **report_clock_gating**

```
*****
Report : clock_gating
Design : top
Version: G-2012.06
```

Date : Sat Aug 4 07:23:00 2012

Clock Gating Summary

Number of Clock gating elements	13
Number of Gated registers	47 (94.00%)
Number of Ungated registers	3 (6.00%)
Total number of registers	50

Clock Gating Multibit Decomposition

	Actual Count	Single-bit Equivalent
Number of Gated Registers		
1-bit	23	23
2-bit	2	4
4-bit	4	16
Total	29	43
Number of Ungated Registers		
1-bit	3	3
2-bit	0	0
4-bit	1	4
Total	4	7
Total Number of Registers		
1-bit	26	26
2-bit	2	4
4-bit	5	20
Total	33	50

The following example shows a report generated with the **-gating_elements** option. The values of STYLE, MIN, MAX, HOLD, SETUP, and OBS_DEPTH for the gating cell are the default values.

```
prompt> report_clock_gating -gating_elements
```

Report : clock_gating
-gating_elements

Design : low_design

Version: 2000.11

Date : Thu Jun 21 18:57:56 2001

Clock Gating Cell Report

Clock Gating Bank : clk_gate_lowout_reg

STYLE = latch, MIN = 3, MAX = 2048, HOLD = 0.00, SETUP = 0.00, OBS_DEPTH = 5

INPUTS :

clk_gate_lowout_reg/CLK = clk

clk_gate_lowout_reg/EN = n_80

OUTPUTS :

clk_gate_lowout_reg/ENCLK = n45

Clock Gating Summary

Number of Clock gating elements	1
Number of Gated registers	4 (66.67%)
Number of Ungated registers	2 (33.33%)
Total number of registers	6

The following example displays additional information about the gated register and its gating cell:

prompt> **report_clock_gating -gated**

Report : clock_gating

-gated

Design : low_design

Version: 2000.11

Date : Thu Jul 26 20:41:33 2001

Gated Register Report

Clock Gating Bank	Gated Register
clk_gate_lowout_reg	lowout_reg[3] lowout_reg[2] lowout_reg[1] lowout_reg[0]

Clock Gating Summary

Number of Clock gating elements	1
Number of Gated registers	4 (66.67%)
Number of Ungated registers	2 (33.33%)
Total number of registers	6

The **-ungated** option displays additional information about ungated registers and the reason they were not gated. If clock gating was inserted with the **insert_clock_gating** command, a brief report is printed, which catches four ungated reasons. But if clock-gating was inserted with the **compile_ultra -gate_clock** command, additional reasons are captured and reported. Also, for some of these reasons there is a "What next" recommendation on how to fix it.

prompt> **report_clock_gating -ungated**

Report : clock_gating

-ungated

Design : low_design

Version: 2000.11

Date : Thu Jul 26 21:02:45 2001

Ungated Register Report

Ungated Register	enable	ctrl	width	removed
lowout_reg[0]	-	-	NO	-
lowout_reg[1]	-	-	NO	-
lowout_reg[2]	-	-	NO	-
lowout_reg[3]	-	-	NO	-
lowstate_reg[0]	-	-	NO	-
lowstate_reg[1]	-	-	NO	-

Clock Gating Summary

Number of Clock gating elements	0
Number of Gated registers	0 (0.00%)
Number of Ungated registers	6 (100.00%)
Total number of registers	6

The following report, displayed after **compile_ultra -gate_clock** command shows that some registers did not meet the minimum bit-width condition for clock gating, while others were explicitly excluded by the user;

prompt> **report_clock_gating -ungated**

Report : clock_gating

-ungated

Design : low_design

Version: 2000.11

Date : Thu Jul 26 21:02:45 2001

Ungated Register Report

Ungated Register	Ungating Reason	What next?

```

-----
lowout_reg[0]      | Minimum bit width not met |
lowout_reg[1]      | Minimum bit width not met |
lowout_reg[2]      | Minimum bit width not met |
lowout_reg[3]      | Minimum bit width not met |
lowstate_reg[0]    | User excluded register   |
lowstate_reg[1]    | User excluded register   |
|                  |                           |
-----

```

Clock Gating Summary

```

-----
| Number of Clock gating elements |    0    |
|                               |         |
| Number of Gated registers      | 0 (0.00%) |
|                               |         |
| Number of Ungated registers    | 6 (100.00%) |
|                               |         |
| Total number of registers      |    6    |
-----

```

The following example shows a report of the physical information for a gating cell with its gated register. The report shows location and distance statistics.

prompt> **report_clock_gating -physical -gating_elements**

```

*****
Report : clock_gating
        -physical
        -gating_elements
Design : timer
Version: 2001.08
Date   : Thu Jul 26 16:43:04 2001
*****

```

Clock Gating Cell Report

Clock Gating Bank : clk_gate_ccr_reg

STYLE = latch, MIN = 3, MAX = 8, OBS_DEPTH = 5

INPUTS :

```

clk_gate_ccr_reg/CLK = pclk
clk_gate_ccr_reg/EN = n_156
clk_gate_ccr_reg/TE = ir[3]

```

OUTPUTS :

```

clk_gate_ccr_reg/ENCLK = n747

```

LOCATION = (52.80,0.00)

Gating group:

Bounding box: (18.40, 0.00)(52.80, 32.00)

Max Distance: 46.98 Min Distance: 17.60 Average Distance: 33.84

Register	Distance	Dist./Avg.
ccr_reg[3]	35.06	103.6%
ccr_reg[5]	32.09	94.8%
ccr_reg[4]	46.98	138.8%
ccr_reg[2]	36.91	109.1%
ccr_reg[1]	34.40	101.7%
ccr_reg[0]	17.60	52.0%

Clock Gating Summary

Number of Clock gating elements	12
Number of Gated registers	92 (90.20%)
Number of Ungated registers	10 (9.80%)
Total number of registers	102

The following example shows a report generated with the **-structure** option for a hierarchical multistage clock gated design with 2 clock domains. In this case the field "Maximum number of stages" is also included in the summary printed at the end.

prompt> **report_clock_gating -structure**

Report : clock_gating

-structure

Design : icg_test

Version: B-2008.09

Date : Tue Aug 5 14:41:11 2008

Clock Gating Structure Summary

Clock	Total Registers	CG Stage	# of Clock Gates	# of Gated Cells
clka	10	1	2	6
		2	1	5
		3	1	2
clkb	10	1	2	6
		2	1	5
		3	1	2

Clock Gating Structure Details

Clock	CG Stage	Gating Element	Fanout	Latency	Gated Cells
clka	1	U1/clk_gate_y_reg_1	3	2.500	U1/y_reg[2]
					U1/y_reg[1]
					U1/y_reg[0]

		U1/clock_gate_y_reg	3	2.500	U1/y_reg[5]
					U1/y_reg[4]
					U1/y_reg[3]
	2	U1/clock_gate_y_reg_0	5	0.700	U1/y_reg[8]
					U1/y_reg[9]
					U1/y_reg[7]
					U1/y_reg[6]
					U1/clock_gate_y_reg
	3	U1/clock_gate_ml	2	0.450	U1/clock_gate_y_reg_1
					U1/clock_gate_y_reg_0
clkb	1	U2/clock_gate_y_reg_1	3	2.500	U2/y_reg[2]
					U2/y_reg[1]
					U2/y_reg[0]
		U2/clock_gate_y_reg	3	2.500	U2/y_reg[5]
					U2/y_reg[4]
					U2/y_reg[3]
	2	U2/clock_gate_y_reg_0	5	0.700	U2/y_reg[9]
					U2/y_reg[8]
					U2/y_reg[7]
					U2/y_reg[6]
					U2/clock_gate_y_reg
	3	U2/clock_gate_ml	2	0.450	U2/clock_gate_y_reg_0
					U2/clock_gate_y_reg_1

Clock Gating Summary

Number of Clock gating elements	8
Number of Gated registers	20 (100.00%)
Number of Ungated registers	0 (0.00%)
Maximum number of stages	3
Total number of registers	20

SEE ALSO

- apply_clock_gate_latency(2)
- compile(2)
- compile_ultra(2)
- remove_clock_gating(2)
- reset_clock_gate_latency(2)
- rewire_clock_gating(2)

```
set_clock_gate_latency(2)
set_clock_gating_objects(2)
set_clock_gating_style(2)
```

report_clock_gating_check

Prints a report of the clock-gating checks.

SYNTAX

```
string report_clock_gating_check  
  [-nosplit]  
  [-significant_digits digits]  
  [instance_list]
```

Data Types

digits integer
instance_list list

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. By default, most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line in the same column.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are reported. Allowed values are from 0 through 13; the default is 2. Using this option overrides the value set by the variable **report_default_significant_digits**.

instance_list

Specifies designs (all gated clock checks in the specified design), clocks, cells, or pins for which to generate a clock-gating report. By default, the report is generated for the current design.

DESCRIPTION

The **report_clock_gating_check** command reports the clock-gating checks and the setup hold values. Information displayed about the specified objects include instance of the cell, enable pin, clock pin, setup/hold time for rise, setup/hold time for fall, user-specified high/low active waveform and clock-gating check attributes. The attributes show from where the clock-gating check was originally defined:

- i** Automatically inferred by Design Compiler
- p** Power Compiler inserted the check

I Defined by the library cell

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example reports user-defined clock-gating checks:

```
prompt> set_clock_gating_check -setup 0.27 -hold 0.38
```

```
prompt> report_clock_gating_check
```

```
*****
Report : clock gating check
Design : top_design
Version: X-2005.09
Date   : Fri Nov 11 14:47:08 2005
*****
```

			Rise	Fall					
Cell	Enable	Clock	Setup	Hold	Setup	Hold	Low/High	Attr	
g1/U1	A	B	0.27	0.38	0.27	0.38	High	i	
g1/U2	A	B	0.27	0.38	0.27	0.38	Low	i	
g1/U3	B	A	0.27	0.38	0.27	0.38	High	i	
g1/U4	A	B	0.27	0.38	0.27	0.38	Low	i	

Disabled:

None

Attr:

i:auto inferred, p:power compiler inserted, l:library cell defined

In the following example, the first command sets clock-gating checks on the design named *test*. The second command sets a clock gating check on the *g1/U2* cell with a non-controlling interval of *high*. The third command disables the clock-gating check on the *g1/U3/B* pin. The fourth command generates a clock-gating check report.

The (*) indicates user override of the non-controlling interval for cell *g1/U2*, which is different from what was determined internally. There are no Power Compiler clock-gating checks in this design. There is one clock-gating check that is disabled on pin *g1/U3/B*.

```
prompt> set_clock_gating_check -setup 0.33 -hold 0.42 test
```

```
prompt> set_clock_gating_check -high -setup 0.21 -hold 0.29 g1/U2
```

```
prompt> set_disable_clock_gating_check g1/U3/B
```

```
prompt> report_clock_gating_check
```

```
*****
Report : clock gating check
Design : test
Version: X-2005.09
```

Date : Fri Nov 11 14:52:18 2005

		Rise		Fall									
Cell	Enable	Clock	Setup	Hold	Setup	Hold	Low/High	Attr					

g1/U1	A	B	0.33	0.42	0.33	0.42	High	i					
g1/U2	A	B	0.21	0.29	0.21	0.29	High (*)	i					
g1/U4	A	B	0.33	0.42	0.33	0.42	Low	i					

Disabled:
g1/U3 B A 0.33 0.42 0.33 0.42 High

Attr:
i:auto inferred, p:power compiler inserted, l:library cell defined

SEE ALSO

- remove_clock_gating_check(2)
- remove_disable_clock_gating_check(2)
- set_clock_gating_check(2)
- set_disable_clock_gating_check(2)
- report_default_significant_digits(3)

report_clock_timing

Reports the timing attributes of clock networks.

SYNTAX

```
status report_clock_timing
  -type report_type
  [-clock clock_list]
  [-from_clock from_clock_list]
  [-to_clock to_clock_list]
  [-to to_list]
  [-from from_list]
  [-setup] | [-hold]
  [-launch] | [-capture]
  [-rise] | [-fall]
  [-min] | [-max]
  [-nworst worst_entries]
  [-greater_than lower_limit]
  [-lesser_than upper_limit]
  [-slack_lesser_than slack_upper_limit]
  [-include_uncertainty_in_skew]
  [-verbose]
  [-show_clocks]
  [-nosplit]
  [-significant_digits digits]
  [-nets]
  [-capacitance]
  [-attributes]
  [-physical]
  [-filter_false_paths]
  [-variation]
  [-scenarios scenario_list]
  [-max]
  [-fall]
  [-capture]
  [-hold]
```

Data Types

<i>report_type</i>	string
<i>clock_list</i>	list
<i>from_clock_list</i>	list
<i>to_clock_list</i>	list
<i>to_list</i>	list
<i>from_list</i>	list
<i>worst_entries</i>	integer
<i>lower_limit</i>	float

```

upper_limit      float
slack_upper_limit float
digits           integer
scenario_list     list

```

ARGUMENTS

-type *report_type*

Specifies the type of report to be generated. Allowed values are as follows:

- **transition** - specifies a transition time report.
- **latency** - specifies a latency report.
- **skew** - specifies a skew report. You cannot use the **-launch**, **-capture**, **-rise**, **-fall**, **-min**, **-max**, and **-lesser_than** options if you specify a skew report. You can use the **-include_uncertainty_in_skew** option only in a skew, interclock_skew, or summary report.
- **interclock_skew** - specifies an interclock skew report. You cannot use the **-launch**, **-capture**, **-rise**, **-fall**, **-min**, **-max**, and **-lesser_than** options if you specify an interclock skew report. You can use the **-include_uncertainty_in_skew** option only in a skew, interclock_skew, or summary report.
- **summary** - specifies a summary report that shows the worst instances of transition time, latency, and skew over the clock networks or subnetworks of interest. You can use only the **-clock**, **-to_list**, **-from_list**, **-include_uncertainty_in_skew**, **-nosplit**, and **-significant_digits** options if you specify a summary report.

For non-summary reports, report entries are ordered with respect to the specified attribute of interest (transition time, latency, or skew). All skews reported are "local" skews. For an explanation of local skew, see the DESCRIPTION section.

-clock *clock_list*

Specifies a list of clock networks to be used in the report. A subreport is produced for every clock in *clock_list*, unless you additionally specify a *to_list* or a *from_list* that has no network intersection with a given clock. In that case, the tool drops these clocks from the *clock_list* and issues a warning. By default, if you do not specify *clock_list*, all clocks in the design that have associated clock networks are used in the report.

-from_clock *from_clock_list*

Specifies a list of clock networks to be used as from-clocks in the current interclock skew report. This option can be used only in an interclock skew report. The report considers every clock in the *from_clock_list*, unless you additionally specify a *from_list* that has no network intersection with a given clock. In that case, the tool drops these clocks from the *from_clock_list* and issues a warning. By default, if you do not specify a *from_clock_list*, all clocks in the design that have associated clock networks are used as from-clocks in the report.

-to_clock *to_clock_list*

Specifies a list of clock networks to be used as to-clocks in the current interclock skew report. This option can be used only in an interclock skew report. The report considers every clock in the *to_clock_list*, unless you additionally specify a *to_list* that has no network intersection with a given clock. In that case, the tool drops these clocks from the *to_clock_list* and issues a warning. By default, if you do not specify a *to_clock_list*, all clocks in the design that have associated clock networks are used as to-clocks in the report.

-to *to_list*

Specifies the list of sequential clock network pins or ports to consider as to-pins in the current report. If any named pin is not the

clock pin of a sequential device (such as a sink pin), the tool replaces that pin with all sequential clock pins in the transitive fanout of the named pin. If there are no sequential clock pins in the pin's transitive fanout, the pin is dropped from the list with a warning message.

For skew reports, the from-to skew sense is defined by the pins in the *from_list* and the *to_list*, respectively. If the *to_list* is not specified, by default all sink pins in the given clock networks are used. Thus, specifying *to_list* reduces the topological scope of the report. For transition time and latency reports, *from_list* and *to_list* are concatenated and treated as a single list. If neither list is specified, *to_list* is assumed to be populated with all sink pins in the given clock networks.

-from *from_list*

Specifies a list of sequential clock network pins or ports to consider as from-pins in the current report. If a named pin is not the clock pin of a sequential device, the tool replaces that pin with all sequential clock pins in the transitive fanout of the named pin. If there are no sequential clock pins in the pin's transitive fanout, the pin is dropped from the list with a warning message.

For skew reports, the from-to skew sense is defined by the pins in the *from_list* and the *to_list*, respectively. If the *from_list* is not specified, by default all sink pins in the given clock networks are used. Specifying the *from_list* reduces the topological scope of the report. For transition time and latency reports, the *from_list* and *to_list* are concatenated and treated as a single list. If neither is specified, the *from_list* is assumed to be populated with all sink pins in the given clock networks.

-setup

Specifies that only the setup data path is to be used in the report, and that the skews, latencies, or transition times reported must correspond to those used by the **report_timing** command in the verification of setup constraints. The **-setup** and **-hold** options are mutually exclusive. If neither option is specified, **-setup** is assumed. This option cannot be used in summary reports.

-launch

Specifies that only pins launching data are to be used in the report, and that latencies or transition times reported must correspond to those used by the **report_timing** command for sequential device clock pins that are launching data. In skew reports, the role is implicit from the from-to sense indicated by the *from_list* and the *to_list*. In all other reports, the **-launch** and **-capture** options are mutually exclusive. If neither option is specified, **-launch** is assumed. This option cannot be used in summary or skew reports.

-rise

Specifies that the active transition is a rising edge for sequential device clock pins in the current report. The **-rise** and **-fall** options are mutually exclusive. If neither option is specified, the active transition at a latch or flip-flop is deduced from the launch or capture role and the behavior of the sequential device itself. This option enables you to answer "what if" questions regarding latency and transition time at sequential device clock pins. This option cannot be used in summary or skew reports.

-min

Specifies that the reports are to be sorted by minimum latencies or transition times. The **-min** and **-max** options are mutually exclusive. If neither option is specified, the reports are sorted in the same manner as described for the **-nworst** option. This option cannot be used in summary or skew reports. To check the quality of a balanced clock tree network, you can generate two latency reports, one with **-min** and one with **-max**.

-nworst *worst_entries*

Specifies the number of worst report entries to be reported per clock domain. The default value is 1. Entries are sorted with respect to the attribute of interest (transition time, latency, or skew) specified with **-type *report_type***.

The worst entries are those most likely to cause a violation. For example, if you request a latency report using **-setup** and **-capture**, the smallest *worst_entries* are listed, sorted in ascending order, because small capture latencies for setup paths are more likely to lead to a violation than large capture latencies. For skew reports, the worst entries always correspond to the largest skews over the specified domain. The above definition of "worst" can be overridden by use of the **-min** or **-max** options. This option cannot be used in summary reports.

-greater_than *lower_limit*

Specifies that only those entries whose attribute value (latency, transition time, or skew) is greater (more positive) than *lower_limit* are shown. This option cannot be used in summary reports.

-lesser_than_upper_limit

Specifies that only those entries whose attribute value (latency, transition time, or skew) is less (more negative) than *upper_limit* are shown. This option cannot be used in summary or skew reports.

-slack_lesser_than_slack_upper_limit

Specifies that only those entries whose slack value is less (more negative) than *slack_upper_limit* are shown. For skew report entries, slack is the worst slack over all paths launched by the from-pin and captured by the to-pin. For transition time or latency report entries, slack is defined as the worst slack of all paths either launched by a transition at the sink pin (if **-launch** is used) or captured by a transition at the sink pin (if **-capture** is used).

Using this filter can greatly increase the runtime of this report. However, when used with **-capture**, the effect on runtime should be small, since the tool is able to make use of cached slack values to avoid expensive recomputation. This option cannot be used in summary reports.

-include_uncertainty_in_skew

Specifies that the user-defined uncertainty between the sequential devices in a launch/capture pair is to be considered a component of skew. This option can be used only in skew and summary reports.

-verbose

Specifies that a more detailed report is to be generated. Instead of a single line per pin, verbose reports trace the entire source-to-sink path through a clock network to show how the final reported attribute (skew, latency, or transition time) was accumulated over the course of the path. This option cannot be used in summary reports.

-show_clocks

Specifies that the launching and capturing clocks are to be shown for every interclock skew entry in the report. This is useful if the *from_clock_list* or the *to_clock_list* contains more than one clock each. This option can only be used in interclock skew reports.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. By default, most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-significant_digits digits

Specifies the number of digits after the decimal point to be displayed for time values in the generated report. Allowed values are 0-13. The default value is determined by the **report_default_significant_digits** variable, whose default value is 2. Use this option if you want to override the default. This option controls only the number of digits displayed, not the precision used internally for analysis. For analysis, the tool uses the full precision of the platform's fixed-precision, floating-point arithmetic capability.

-nets

Shows nets in verbose path traces. The default is not to show nets. To show the delay for the nets, use the **-input_pins** option. This option is similar to its counterpart in the **report_timing** command.

-capacitance

Shows the total (lumped) capacitance in verbose path traces. The default is not to show capacitance. For each driver pin, the total capacitance driven by the driver is displayed in a column preceding both incremental path delay and transition time. When **-nets** is specified, the capacitance is printed on the lines with nets instead of the lines with driver pins. This option is similar to its counterpart in the **report_timing** command.

-attributes

Shows in verbose path traces the attributes specified in the **timing_report_attributes** variable. The current set of attributes supported are **dont_touch**, **dont_use**, **map_only**, and **size_only** for cells, and **dont_touch** and **ideal_net** for nets. This option is similar to its counterpart in the **report_timing** command.

-physical

Shows the locations of the pins and the capacitive loads for the pins and nets in verbose path traces. The loads are displayed as a pair of values of which the first is the wire capacitance of the net and the second value is the total capacitance driven by the driver. If the pin location cannot be determined, the cell location is displayed, with the coordinates in microns. This option is similar to its counterpart in the **report_timing** command.

-filter_false_paths

Causes the command to eliminate false paths from consideration during calculation of local register-to-register skew, resulting in more accurate (less pessimistic) reporting of skew. The default behavior is to not eliminate false paths from consideration.

-variation

Reports the mean and standard deviation of the statistical time increment for each source-to-sink path element in a **-verbose** report. This option works only when parametric on-chip variation (POCV) analysis has been enabled by setting the **timing_pocvm_enable_analysis** variable to true. By default, statistical parameters are not reported.

-scenarios *scenario_list*

Reports clock timing for a specified set of scenarios and skips reporting of inactive scenarios.

-max

Specifies that the reports are to be sorted by maximum latencies or transition times. The **-min** and **-max** options are mutually exclusive. If neither option is specified, the reports are sorted in the same manner as described for the **-nworst** option. This option cannot be used in summary or skew reports. To check the quality of a balanced clock tree network, you can generate two latency reports, one with **-min** and one with **-max**.

-fall

Specifies that the active transition is a falling edge for sequential device clock pins in the current report. The **-rise** and **-fall** options are mutually exclusive. If neither option is specified, the active transition at a latch or flip-flop is deduced from the launch or capture role and the behavior of the sequential device itself. This option enables you to answer "what if" questions regarding latency and transition time at sequential device clock pins. This option cannot be used in summary or skew reports.

-capture

Specifies that only pins capturing data are to be used in the report, and that the latencies or transition times reported must correspond to those used by the **report_timing** command for sequential device clock pins that are capturing data. In skew reports, the role is implicit from the from-to sense indicated by the *from_list* and the *to_list*. In all other reports, the **-launch** and **-capture** options are mutually exclusive. If neither option is specified, **-launch** is assumed. This option cannot be used in summary or skew reports.

-hold

Specifies that only the hold data path is to be used in the report, and that the skews, latencies, or transition times reported must correspond to those used by the **report_timing** command in the verification of hold constraints. The **-setup** and **-hold** options are mutually exclusive. If neither option is specified, **-setup** is assumed. This option cannot be used in summary reports.

DESCRIPTION

This command generates a report of clock timing information for the current design.

Several reporting options allow you to examine skew, latency, and transition time attributes of a specified clock network or subnetwork at various levels of generality. By default, the report displays the values of these attributes only at sink pins (that is, the clock pins of sequential devices) of the clock network. Use the **-verbose** option to display source-to-sink path traces. If you specify several clock domains, **report_clock_timing** generates a separate subreport for each domain.

At the highest level of abstraction is the summary style report, which provides only a list of maxima and minima of the skew, latency, and transition time attributes over the given networks. At a lower level of abstraction are the transition, latency, and skew type reports, called list style reports, in which you can sort, filter, and display the worst set of sink pins in the given network with respect to a single attribute of interest. For skew reports, each report entry is a pair of sink pins and their relative skew. For transition time or latency reports, each entry corresponds to a single sink pin. The lowest level of abstraction is provided by the verbose mode, which replaces every sink pin in a list style report with a corresponding source-to-sink path trace.

In both summary and list style reports, the right column is an attribute column. Corresponding to each sink pin, the character symbols in this column indicate the following:

Symbol	Meaning
-----	-----
w	Worst-case operating condition
b	Best-case operating condition
r	Rising transition
f	Falling transition
p	Propagated clock to this pin
i	Clock inversion to this pin
e	Exception on this pin
-	Launching transition
+	Capturing transition

In verbose mode, back-annotations on path elements in the timing path are indicated by a character symbol. For definitions of these character symbols, see the man page for the **report_timing** command.

Skews reported by **report_clock_timing** are local skews only. Local skew exists from one sink pin to another only as long as the associated sequential devices are connected via a data path in the appropriate from-to sense. Note that even if all data paths between the sequential devices are false because of user-defined exceptions, the skew still qualifies as local skew if the devices are connected topologically.

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following example shows a typical summary style report:

```
prompt> report_clock_timing -type summary
```

```
*****
Report : clock timing
      -type summary
      -nworst 1
Design : xor_testcase
Version: W-2004.12
Date   : Tue Nov 20 06:17:33 2004
*****
```

Clock: CK1

```

-----
Maximum setup launch latency:
  flop9/CP                3.08    rp-+

Minimum setup capture latency:
  or2_3/B                 1.15    fpi-+

Minimum hold launch latency:
  or2_3/B                 1.15    fpi-+

Maximum hold capture latency:
  flop9/CP                3.08    rp-+

Maximum active transition:
  or2_3/B                 0.20    fpi-+

Minimum active transition:
  or2_3/B                 0.09    fpi-+

Maximum setup skew:
  flop9/CP                rp-+
  flop10/CP               0.87    rp-+

Maximum hold skew:
  flop9/CP                rp-+
  flop10/CP              -0.21    rp-+
-----

```

The following example displays the five worst setup skews in the clock network of CLK1, taking uncertainty into account:

```

prompt> report_clock_timing -clock CLK1 -type skew -setup \
-nworst 5 -include_uncertainty_in_skew

```

Report : clock timing

-type skew

-nworst 5

-setup

-include_uncertainty_in_skew

Design : multi_domain

Version: W-2004.12

Date : Tue Nov 20 06:49:19 2004

Clock: CLK1

```

Clock Pin          Latency  Uncert  Skew
-----
f2_2/CP            6.11          rp-+
f2_1/CP            2.01    0.11    4.21    fpi-+

l3_2/G             4.10          rpi-
f1_2/CP            1.00    0.22    3.32    rpi-+

l2_2/G             4.11          rp-

```

f1_2/CP	1.00	0.12	3.23	rpi-+
f2_2/CP	6.11			rp-+
f3_3/G	3.01	0.11	3.21	rp-
f1_3/G	5.11			rp-
f2_1/CP	2.01	0.11	3.21	fp-+

The following example displays the five worst launching latencies for hold paths in the clock network of CLK2:

```
prompt> report_clock_timing -clock CLK2 -hold -launch -nworst 5 \
-type latency
```

```
*****
```

```
Report : clock timing
```

```
-type latency
```

```
-launch
```

```
-nworst 5
```

```
-hold
```

```
Design : multi_domain
```

```
Version: W-2004.12
```

```
Date : Tue Nov 20 06:55:56 2004
```

```
*****
```

```
Clock: CLK2
```

Clock Pin	--- Latency ---				
	Trans	Source	Network	Total	
f1_2/CP	0.04	0.00	1.00	1.00	rpi-+
f1_3/CP	0.00	0.01	1.00	1.01	fp-+
f2_1/CP	0.01	0.01	2.00	2.01	rp-+
f1_1/CP	0.00	0.00	3.00	3.00	rpi-+
f3_2/CP	0.00	0.00	3.00	3.00	fpi-+

The following example demonstrates how to request a verbose report showing the worst local skew from f2_2/CP to any other sink pin:

```
prompt> report_clock_timing -type skew -verbose -from f2_2/CP
```

```
*****
```

```
Report : clock timing
```

```
-type skew
```

```
-verbose
```

```
-nworst 1
```

```
-setup
```

```
Design : multi_domain
```

```
Version: W-2004.12
```

```
Date : Tue Nov 20 07:00:56 2004
```

```
*****
```

```
Clock: CLK1
```

```
Startpoint: f2_2 (rising edge-triggered flip-flop clocked by CLK1)
```

```
Endpoint: f2_1 (rising edge-triggered flip-flop clocked by CLK1)
```


Point	Trans	Incr	Path

clock source latency		0.11	0.11
clk2 (in)	0.00	0.00	0.11 r
az_1/Z (B1I)	0.09	1.00 H	1.11 r
az_2/Z (B1I)	0.13	1.00 H	2.11 r
bf2_2_1/Z (B1I)	0.02	1.00 H	3.11 r
if2_2_1/Z (IVA)	0.44	1.00 H	4.11 f
bf2_2_2/Z (B1I)	0.01	1.00 H	5.11 f
if2_2_2/Z (IVA)	0.13	1.00 H	6.11 r
f2_2/CP (FD1)	0.13	0.00	6.11 r
startpoint clock latency			6.11
clock source latency		0.01	0.01
clk2 (in)	0.00	0.00	0.01 r
az_1/Z (B1I)	0.02	1.00 H	1.01 r
az_3/Z (B1I)	0.01	1.00 H	2.01 r
f2_1/CP (FD1)	0.01	0.00	2.01 r
endpoint clock latency			2.01

startpoint clock latency			6.11
endpoint clock latency			-2.01

skew		4.10	

The following example traces the two worst launch latencies for hold paths in the clock network of CLK1:

```
prompt> report_clock_timing -type latency -hold -verbose \
-nworst 2 -clock CLK1
```

```
*****
```

Report : clock timing

```
-type latency
-verbose
-launch
-nworst 2
-hold
```

Design : multi_domain

Version: W-2004.12

Date : Tue Nov 20 07:14:28 2004

```
*****
```

Clock: CLK1

Endpoint: f1_2 (rising edge-triggered flip-flop clocked by CLK1')

Point	Trans	Incr	Path

clock source latency		0.00	0.00
clk1 (in)	0.00	0.00	0.00 f
if1_2_1/Z (IVA)	0.04	1.00 H	1.00 r
f1_2/CP (FD1)	0.04	0.00	1.00 r
total clock latency			1.00

Endpoint: f1_3 (rising edge-triggered flip-flop clocked by CLK1')

Point	Trans	Incr	Path

clock source latency		0.01	0.01
clk1 (in)	0.00	0.00	0.01 r
bf1_3_1/Z (B1I)	0.00	1.00 H	1.01 r
f1_3/CP (FD1)	0.00	0.00	1.01 r
total clock latency		1.01	

The following example makes use of a slack filter to display the worst three skews between latches whose latch-to-latch paths are violating:

```
prompt> report_clock_timing -type skew -nworst 3 \
        -slack_lesser_than 0.0
```

```
*****
```

Report : clock timing

-type skew

-slack_lesser_than 0.00

-nworst 3

-setup

Design : multi_domain

Version: W-2004.12

Date : Fri Dec 14 09:11:05 2004

```
*****
```

Clock: CLKA

Clock Pin	Latency	CRP	Skew	Slack

f2_2/CP	6.01			rp-+
f2_1/CP	2.11	-0.03	3.87	-1.49 rp-+
l2_2/G	4.01			rp-
f1_2/CP	1.10	-0.21	2.70	-6.64 rpi-+
l1_3/G	5.01			rp-
f2_1/CP	2.11	-0.32	2.58	-1.63 rp-+

The following example requests the two largest setup skews for paths both launched and captured by any clock networks in the \$my_clocks variable:

```
prompt> report_clock_timing -type interclock_skew \
        -from_clock $my_clocks -to_clock $my_clocks \
        -nworst 2 -include_uncertainty_in_skew -show_clocks
```

```
*****
```

Report : clock timing

-type interclock_skew

-nworst 100

-setup

-include_uncertainty_in_skew

-show_clocks

Design : my_design

Version: W-2004.12

Date : Thu May 2 10:55:20 2004

```
*****
```

Number of startpoint pins : 907
 Number of endpoint pins : 907
 Number of startpoint clocks : 5
 Number of endpoint clocks : 5

Clock Pin	Latency	Uncert	Skew

orig_clk			
orig_if/ram_pdp1/FFB35/CK	1016.72		rp-+
dcd_ram/ram_clk			
dcd_ram/dcd_ram/RAM/CKRB	149.60	195.00	1062.12 rp-+
comp_clk			
comp_if/c_port_if/edge_trig_reg/CK	1400.42		rp-+
comp_clk			
comp_if/c_port_if/posi/iq_reg/CK	1159.09	199.00	440.34 rp-+

SEE ALSO

report_clock(2)
 report_timing(2)
 set_clock_uncertainty(2)
 report_default_significant_digits(3)
 timing_pocvm_enable_analysis(3)
 timing_remove_clock_reconvergence_pessimism(3)

report_clock_tree

Reports the structural and timing characteristics of a compiled clock tree.

SYNTAX

```
status report_clock_tree
  [-clock_trees clock_tree_list]
  [-summary]
  [-structure]
  [-drc_violators]
  [-settings]
  [-exceptions [-show_all_sinks]]
  [-from from_list | -to to_list]
  [-operating_condition condition]
  [-level_info]
  [-high_fanout_net net_or_pin_list
   | -premesh
   | -postmesh]
  [-nosplit]
  [-all_drc_violators]
  [-partial_structure_within_exceptions]
  [-skew_group skew_groups_string]
  [-histogram_transition file_name
   | -histogram_capacitance file_name
   | -histogram_rcdelay file_name
   | -histogram_fanout file_name
   | -histogram_rcdelay_to_sink file_name]
  [-local_skew
   | -histogram_local_skew file_name]
  [-local_skew_skip_icg]
  [-nworst number_of_pin_pairs]
  [-launch_pins launch_pin_list]
  [-capture_pins capture_pin_list]
  [-number_of_bins bin_number]
  [-sink_group
   | -sink_group_ignore_cts_exceptions
   | -sink_group_timing_relationship_limit limit
   | -sink_group_sort_by clock | sink_group
   | -sink_group_sort_order ascending | descending
   | -sink_group_timing_relationship_file file_name
   | -sink_group_detail_file file_name
   | -sink_group_ignore_buf_inv
   | -sink_group_ignore_icg]]
  [-scenarios list_of_scenarios]
  [-interclock_timing]
```

Data Types

```

clock_tree_list    list
from_list         list
to_list           list
condition         string
net_or_pin_list    list
skew_groups_string string
file_name         string
number_of_pin_pairs integer
launch_pin_list    list
capture_pin_list   list
bin_number        integer
limit             integer
list_of_scenarios  list

```

ARGUMENTS

-clock_trees *clock_tree_list*

Reports only the clock trees specified in *clock_tree_list*. By default, the command reports all currently defined clock trees.

This option and the **-scenarios** option are mutually exclusive; you can use only one.

-summary

Prints a summary table for the clock trees. The summary table includes the following columns:

- Clock
The name of the clock tree.
- Sinks
The number of sinks in the clock tree.
- CTBuffers
The number of clock tree cells inserted by clock tree synthesis.
- ClkCells
The total number of cells in the clock tree, including preexisting gates.
- Skew
The clock tree skew.
- LongestPath
The longest path delay seen by the clock.
- Total DRC
The total number of DRC violations in the clock tree.
- BufferArea
The area of all the CTBuffers in the clock tree.

The **-summary** option can be used with the **-local_skew** option or combined with the **-local_skew** and **-local_skew_skip_icg** options. When you combine **-summary** with **-local_skew -local_skew_skip_icg**, the following columns are added to the report:

- LocalSkew
The Local skew value of the clock.

- **LLatency**
The largest latency value.
- **SLatency**
The smallest latency value.

-structure

Prints the complete structure of the clock tree. This report traverses through exceptions such as explicit float pins and implicit and explicit exclude pins.

-drc_violators

Lists the design rule checking (DRC) violations that occur in the clock trees up to the endpoints (default sinks or don't touch subtrees), including violations beyond exceptions on stop pins, exclude pins, and float pins but excluding violations on don't buffer nets, don't touch subtrees, nets inside interface logic models, and nets connected to boundary cells or pad cells.

-settings

Prints the clock tree settings specified by the **set_clock_tree_options** command, and references available for buffering, design rule checking (DRC) constraints, the list of available routing layers, and so on.

-exceptions

Prints a detailed list of clock tree exceptions.

-show_all_sinks

Reports all default sinks in the exception report when used with the **-exceptions** option.

-from *from_list*

Specifies the list of pin names whose fanout clock tree is reported.

-to *to_list*

Specifies the list of pin names to which the clock paths are reported. The specified pins must be clock tree sink pins. This option can be used with the **-operating_condition** and **-clock_trees** options.

-operating_condition *condition*

Reports the clock tree based on the specified condition.

-level_info

Reports information on the clock tree by level.

-high_fanout_net *net_or_pin_list*

Reports a tree built by the clock tree synthesis based high-fanout net synthesis engine. This option can be used alone or in combination with the following options:

- **-summary**
- **-structure**
- **-level_info**
- **-drc_violators**
- **-operating_condition**
- **-nosplit**

-premesh

Reports the clock network that has a clock mesh, starting from the clock root to the mesh driver inputs. This option treats the mesh driver inputs as stop pins and ignores the portion of the clock tree beneath the mesh. You can run this option as a standalone process or in combination with the following options:

- **-clock_tree**
- **-summary**
- **-structure**
- **-exceptions**
- **-drc_violators**
- **-operating_condition**
- **-nosplit**

-postmesh

Reports the clock network that has a clock mesh, starting from the inputs of the clock drivers that are the load pins of the clock mesh. This option does not report the clock network from the clock root to the clock mesh. You can run this option as a standalone process or in combination with the following options:

- **-clock_tree**
- **-summary**
- **-structure**
- **-exceptions**
- **-drc_violators**
- **-operating_condition**
- **-nosplit**

When you specify this option by itself, the command reports the longest path, the shortest path, and the skew among all subtrees, starting from the load pins of the clock mesh. In addition, the clock mesh is treated as ideal so that every point of the clock mesh has the same arrival time. Using this option with other options is equivalent to using the **-from load_pins_of_the_clock_mesh** option.

-nosplit

Prevents line splitting to allow ease of use to create scripts to extract information from the report output. By default, the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-all_drc_violators

Forces the **report_clock_tree** command to list all DRC violations in the clock trees, including violations beyond exceptions and violations on don't touch subtrees but excluding violations on don't buffer nets, nets inside interface logic models, and nets connected to boundary cells or pad cells.

-partial_structure_within_exceptions

Prints the part of the clock tree structure that is within explicit exclude and float pin exceptions. The clock structure beyond explicit exclude and float pin exceptions is not printed. Use the **-structure** option to print the complete clock structure.

-skew_group skew_groups_string

Reports the clock skew information only for the specified skew groups. You can separate each skew group with a space in the specified *skew_groups_string*. The default skew group name is **default**. For each specified skew group, the skew information is printed for all clocks that propagate to the skew group. You can only use this option with **-clock_trees** and **-summary**. If you did

not specify the **-skew_group** option but ran the **commit_skew_group** command successfully, the clock skew information of all skew groups is reported.

-histogram_transition *file_name*

Reports the clock pin transition delay information to the specified file in histogram format.

-histogram_capacitance *file_name*

Reports the clock pin capacitance information to the specified file in histogram format.

-histogram_rcdelay *file_name*

Reports the RC delays between clock drivers and load pairs to the specified file in histogram format.

RC-delay histograms must be reported after clock tree synthesis gets valid delay data on clock nets.

-histogram_fanout *file_name*

Reports clock pin fanout numbers to the specified file in histogram format.

-histogram_rcdelay_to_sink *file_name*

Reports the RC delays per sink between clock drivers and each valid sink to the specified file in histogram format.

RC-delay histograms must be reported after clock tree synthesis gets valid delay data on clock nets.

-local_skew

Reports local skews for clock sink pairs with data dependencies. Local skew is the worst-case clock arrival time difference between a pair of registers with data dependency. The local skew report prints out a certain number of the worst local-skew clock pins, as controlled by the **-nworst** option.

The **-local_skew** option can be combined with **-summary**. When you combine these options with the **-local_skew_skip_icg** option, three additional columns (local skew and the largest and smallest local skew of each clock), are added to the summary report. This usage is supported only when you use the **-local_skew_skip_icg** option with the **-local_skew** option.

You can choose to have a histogram report by using the **-histogram_local_skew** option. You use the **-clock**, **-launch_pins**, and **-capture_pins** options to select which clock pin pairs to include in the local skew report.

-histogram_local_skew *file_name*

Reports local skew to the specified file in histogram format.

-local_skew_skip_icg

Skips integrated clock gating cells in the local skew report.

-nworst *number_of_pin_pairs*

Limits the number of clock pin pairs in the local skew report. Only the specified number of pin pairs with the worst local skew are included in the report. If a histogram is enabled for the local skew report, the histogram report file includes all local skew pin pairs, regardless of the **-nworst** option. By default, the value of **-nworst** is 1.

-launch_pins *launch_pin_list*

Includes only the pin pairs with the specified launch pins in the local skew report.

-capture_pins *capture_pin_list*

Includes only the pin pairs with the specified capture pins in the local skew report.

-number_of_bins *bin_number*

Specifies the number of bins that are used to generate the histogram report. Use this option when you specify at least one of the histogram reporting options: **-histogram_transition**, **-histogram_capacitance**, **-histogram_rcdelay**, **-histogram_fanout**, **-histogram_rcdelay_to_sink**, and **-histogram_local_skew**. By default, the number of bins is 10.

-sink_group

Enables the sink group reporting feature.

This option works only with the **-clock_trees** option and the **-sink_group_*** options. If you use any other options, they are either ignored or the tool issues an error message.

-sink_group_ignore_cts_exceptions

Ignores the clock tree synthesis exceptions, except for nonstop exceptions, when traversing the clock tree to find the sink groups.

This option is ignored if you do not also use the **-sink_group** option.

-sink_group_timing_relationship_limit *limit*

Reports sink groups that have timing relationships with other sink groups that are less than or equal to the specified limit.

If you set the limit to zero, the command reports only sink groups without any timing relationship.

By default, the limit is infinite and the command reports all sink groups with timing relationships.

This option is ignored if you do not also use the **-sink_group** option.

-sink_group_sort_by clock | sink_group

Sorts the first-level text output either by clock or by sink group. When you specify **clock**, the sort key is the sink number under the clock. When you specify **sink_group**, the sort key is the timing relationship number. A tie is broken by the clock or sink group name.

By default, the sink group report is sorted by clock.

This option is ignored if you do not also use the **-sink_group** option.

-sink_group_sort_order ascending | descending

Specifies whether the sink group report is sorted in ascending or descending order.

By default, the sink group report is sorted in ascending order.

This option is ignored if you do not also use the **-sink_group** option.

-sink_group_timing_relationship_file *file_name*

Writes the timing relationships of the sink pairs across the whole design; without considering any sink group information. Depending on the design, the file size could be huge, so use this option with caution.

This option is ignored if you do not also use the **-sink_group** option.

-sink_group_detail_file *file_name*

Writes detailed sink group information, such as a detailed list of clocks and sinks, to the specified file.

This option is ignored if you do not also use the **-sink_group** option.

-sink_group_ignore_buf_inv

Treats buffer and inverter cells as transparent cells during analysis if there is no clock or generated clock driving them.

This option is ignored if you do not also use the **-sink_group** option.

-sink_group_ignore_icg

Treats integrated clock gating cells as transparent cells during analysis if there is no clock or generated clock driving them.

This option is ignored if you do not also use the **-sink_group** option.

-scenarios *list_of_scenarios*

Specifies the list of active scenarios that the command reports. By default, the command reports only the current scenario.

The **-scenarios** and the **-clock_trees** options are mutually exclusive; you can use only one.

-interclock_timing

Allows you to report interclock timing relationships, which includes worst negative slack (WNS), cumulative negative slack (CNS), the number of violating paths (NVP), and local skew for the WNS path.

If you specify this option, the tool ignores any other options you specify.

DESCRIPTION

The **report_clock_tree** command obtains information on clock trees. The command reports the structural and timing characteristics of a compiled clock tree.

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

In the following example, the **report_clock_tree** command is run on a previously-compiled clock tree called CLK1, and the reported results are based on a clock route estimator:

```
prompt> report_clock_tree -clock_trees CLK1
```

In the following example, the **report_clock_tree** command is run on a precompiled clock tree named CLK1. The report includes a listing of all logical DRC violations and a report of clock tree settings.

```
prompt> report_clock_tree -clock_trees CLK1 \
-drc_violators -settings
```

In the following example, the **report_clock_tree** command is run on a precompiled clock tree named CLK1. The report includes hierarchical level information on the clock tree netlist structure, the distribution clock tree delays, DRC violations, and all exceptions.

```
prompt> report_clock_tree -clock_trees CLK1 \
-structure -drc_violators -exceptions
```

In the following example, the **report_clock_tree** command is run on a precompiled clock tree named clk. The report includes a listing of all clock tree exceptions in the clock tree.

```
prompt> report_clock_tree -clock_trees clk -exceptions
```

```
*****
Report : clock tree
Design : top
Version: Y-2006.06
Date   : Fri Sep 15 17:58:35 2006
*****
```

```
===== Clock Tree Exception Pins =====
```

Clock Tree Exceptions Summary

```
=====
```

```
1. Clock: CLK
  Clock root: clk
  Clock net: clk
  Total sinks: 1
  Explicit ignore pins: 0
  Explicit sync pins: 0
  Implicit ignore pins: 1
  Default sink pins: 1
  Explicit nonstop pins: 0
  Implicit nonstop pins: 0
  Dont touch subtree pins: 0
  Dont buffer nets: 0
  Dont size cells: 0
```

```
=====
```

Clock Tree Exceptions

```
=====
```

Legends Used

```
-----
```

```
(P) = Default sink pin
(F) = Explicit stop/float pin
(I) = Implicit ignore pin
(E) = Explicit ignore pin
(J) = Implicit nonstop pin
(T) = Explicit nonstop pin
(D) = Dont touch subtree pin
(B) = Dont buffer net
(S) = Dont size cell
```

```
1. Clock: CLK
  Clock root: clk
  Clock net: clk
  Total sinks: 1
  Explicit ignore pins: 0
  Explicit sync pins: 0
  Implicit ignore pins: 1
    (I) s/clk
  Default sink pins: 1
  Explicit nonstop pins: 0
  Implicit nonstop pins: 0
  Dont touch subtree pins: 0
```

Dont buffer nets: 0
Dont size cells: 0

=====

In the following example, the **report_clock_tree** command is run on a previously compiled clock tree called CLK1, and two histogram report files are generated. The histogram report of transition delay of all clock pins in clock tree CLK1 is written to the tran_hist.rpt file. The histogram report of RC-delay between all the driver-load pin pairs in clock tree CLK1 is written to the rc_hist.rpt file. The histogram reports are generated by using the default bin number value of 10.

```
prompt> report_clock_tree -clock_trees CLK1 \
-histogram_transition tran_hist.rpt \
-histogram_rcdelay rc_hist.rpt
```

In the following example, the **report_clock_tree** command generates a sink group report.

```
prompt> report_clock_tree -sink_group \
-sink_group_detail_file a
```

Collecting setup paths:

... 20% ... 40% ... 60% ... 80% ... 100%

Done.

Saving sink_group timing relationship detail into file a.pair ...

Saving sink_groups information into file a.detail ...

Legends:

(SG) = total number of sink_groups which belong to this clock

(S) = total number of sinks from all of the sink_groups which belong to this clock

(S1) = total number of sinks of current sink_group

(A1) = total timing relationship number associated with sinks of current sink_group

(F1) = timing relationship number starting from any sink of current sink_group

(T1) = timing relationship number ending to any sink of current sink_group

(A2) = total timing relationship number associated with sinks of the two sink_groups

(F2) = timing relationship number starting from current sink_group to another sink_group

(T2) = timing relationship number starting from another sink_group to current sink_group

Format:

<clock_name> (clock) SG / S

<current_sink_group_name> : <driver_name> (S1) A1 = F1 + T1

<group_with_timing_relationship> A2 = F2 + T2

clk1 (clock) 2 / 16

iclk1 : U28/Z (8) 2 = 0 + 2

gclk1 : U26/Z 2 = 0 + 2

gclk1 : U26/Z (8) 5 = 3 + 2

iclk1 : U28/Z 2 = 2 + 0

gclk2 : U27/Z 3 = 1 + 2

clk2 (clock) 1 / 64

gclk2 : U27/Z (64) 3 = 2 + 1

gclk1 : U26/Z 3 = 2 + 1

1

In the following example, the **report_clock_tree** command generates a interclock_timing report.

```
prompt> report_clock_tree -interclock_timing
```

Report : clock tree
Design : Top_Create
Scenario(s): normal_mode
Version: G-2012.06-ALPHA3
Date : Fri Mar 9 12:18:01 2012

===== Report for scenario (normal_mode)=====

From Clock	To Clock	WNS	CNS	NVP	Local skew(WNS path)
=====	=====	=====	=====	=====	=====
CLK1	CLK2	-1.0	100.1	245	0.1
CLK1	CLK1	-0.5	20.3	100	0.3

SEE ALSO

report_collection

Output a tabular report of attribute values for elements in a collection.

SYNTAX

report_collection

```
collection  
[-type report_type]  
[-header header_type]  
[-max_rows value_count]  
[-columns column_specification]  
[-nosplit]
```

Data Types

<i>collection</i>	string
<i>report_type</i>	list
<i>header_type</i>	string
<i>value_count</i>	int

ARGUMENTS

collection

Specifies the collection on which to report. The collection may be homogeneous or heterogeneous. The report comprises a header with information about the report such as the application name and version and the date and a tabular report on attribute values for elements of the collection. The tabular report content is determined by the of report requested with the *-type* option. The attributes on which the report is generated is determined by the argument to the *-columns* option.

-type report_type

This option accepts an argument of a list of valid report type names. A report type is one of the following values: "values" | "statistics" | "distribution". If the option is omitted, the default report type is "values". The report types may be combined in a list to output multiple reports in one run. Each invocation of the command produces a single report header output unless it is suppressed with the *-header* option.

The "values" report is a tabular output of attribute values for the collection elements, with one row of values per collection element. The report columns comprises formatted attribute values for the objects.

The "statistics" report type outputs the following statistical data about the attribute values

- * Minimum, lower quartile, median upper quartile, and maximum values for numeric attributes.
- * The mean and standard deviation for numeric attributes.
- * Number of null attribute values for each attribute.
- * Number of valid attribute values for each attribute.
- * Number of unique attribute values for each attribute.

The "distribution" report type outputs a header section with information about the report, followed by the following statistical values about the data.

* Number of null attribute values for each attribute. * Number of valid attribute values for each attribute. * Number of unique attribute values for each attribute. * Distribution of values with number of occurrences of each attribute value..

If both "statistics" and "distribution" reports are requested, the overlapping portion of the two reports is output only once.

-header *header_type*

This option accepts one of the following valid argument values: "standard" | "none". If the option is omitted, the default header type is "standard". The "standard" header consists of the report name, object type, product name, product version, and the report date. Precise object types are always shown for "statistics" and "distribution" type reports. For "values" report, it is shown if the collection is homogenous or small. If the collection is heterogeneous and large, the object type shown for "values" report is "heterogeneous".

-max_rows *value_count*

This option indicates how many rows of data to include in the report. This number indicates the number of collection elements on which the report is produced. If the **collection** is very large, you may want to use the **-max_rows** option to limit the size of your report. Number of lines in the report may be greater than the *-max_rows* argument value if some data rows require multiple lines to output.

-columns *column_specification*

Indicates the set of attributes on which to report and their order in the report. At a minimum, *column_specification* is a list of attribute names of elements of the collection.

The special attribute name "object_class" may be used to include the element object class names, such as "cell" or "net", in the report output.

If the option is omitted, then the object full names and object class (or type) names are output by default.

Each attribute name may be followed by a list of token-value pairs to indicate report column formatting directives. The value tokens for formatting are *width*, *justify*, *label*, and *fit*. All column formatting specifications are optional.

width token is followed by a positive integer to value indicate the desired width of the column in number of characters. If not given, then the attribute value is queried for the first 100 elements of the collection and the longest value string determines the width of the column.

justify is followed by one of "left" or "right" to indicate the value's alignment in the column. If not given, then numeric attributes are justified right and other attributes are justified left.

By default, the column headers show the name of the attribute. *label* is followed a string to display in the report column header in place of the attribute name.

fit is followed by one of the following valid argument values: "split" | "none" | "truncate" | "wrap" | "elide_left" | "elide_center" | "elide_right". This option indicates how to format an attribute value that is too long to fit in the column's width. If option is omitted, the default formatting is "split". However, if the option *-nosplit* is given, then the default formatting becomes "none". Explicit *fit* values given in a column specification overrides the default set by *-noSplit*.

The "split" formatting inserts a newline after the value which is too long and continues the tabular row on the next line, justifying the next value to the correct column.

The "none" formatting does nothing to reformat a value that is too long, continuing with the next column value. Therefore, the remainder of the tabular row will not be aligned with column headers.

The "truncate" formatting truncates the value to the width of the column, discarding the remainder of the value.

The "wrap" formatting truncates the value to the column and continues the rest of the tabular row on the same row. The remainder of the value is output on the following line(s), taking as many lines as is needed to finish outputting the rest of the value.

The "elide_left" formatting truncates the beginning of the value and adds the elision string "..." at the beginning of the value.

The "elide_center" formatting omits the middle of the value and inserts the elision string "..." at the center of the value.

The "elide_right" formatting truncates the end of the value and adds the elision string "..." at the end of the value.

-nosplit

This option indicates that the default column formatting used for a value is that is too long for the column width is "none", rather than the default "split". See the -columns option description below for more details.

DESCRIPTION

This command outputs a tabular report of attribute values for elements of the given collection. The attribute values in the report are determined by the list of attributes given as the **-columns** option argument. The columns in the tabular report will be ordered by order of attributes in the list. The command **list_attributes** may be called to see a list of attributes defined for an object class.

The **-column** option also accepts additional optional column formatting specifications in the attribute list. You may specify the header label and the width for each column, how to justify the value, and how to fit a value in a column when the value is too long to fit within the column width.

If the collection is large, the report size may be limited by indicating a **-max_rows** value. The commands **filter_collection** and **sort_collection** may be called to filter and sort a collection based on some criteria prior to creating a report for the collection elements.

Statistical and value distribution reports on the collection elements may be produced using the *-type* option arguments "statistics" and "distribution", respectively. Report types may be combined in a list.

EXAMPLES

The following example from IC Compiler II creates a statistical report on a collection of cells along with a value distribution output.

```
prompt> set cells [get_cells -hierarchical *]
prompt> report_collection $cells -type {statistics distribution}
```

The following example, also from IC Compiler II, creates a report on a collection of cells with the full_name, area, and number_of_pins attribute values.

```
prompt> set cells [get_cells U*]
prompt> report_collection $cells -columns {full_name area number_of_pins}
```

The next example creates the same report but limits the width of the full_name column to 26 characters, designating "elide_center" formatting for full_name values that are longer than 26 characters.

```
prompt> report_collection $cells -columns \
    {{full_name {width 26} {fit elide_center}}} \
    area number_of_pins}
```

The next examples limits the output to the first 10 objects in the collection.

```
prompt> report_collection $cells -columns \
    {{full_name {width 26} {fit elide_center}}} \
    area number_of_pins} -max_rows 10
```


The next example first sorts the original collection of cells by the area attribute value in descending order, limiting the resulting collection to the top 10 area values. The example then creates a report for the resulting collection.

```
prompt> set sorted_cells [sort_collection -dictionary \  
    $cells {area- full_name} -limit 10]  
prompt> report_collection $sorted_cells -columns \  
    {{full_name {width 26} {fit elide_center}}} \  
    area number_of_pins}
```

The next example reports on a sorted collection as in the above example but further limits the report to the first 10 objects in the collection. Note that this command may produce a report that is different from the above example. Limiting the sort to the first 10 values may have produced a collection with more than 10 objects since multiple objects may share the same area value.

```
prompt> set sorted_cells [sort_collection -dictionary \  
    $cells {area- full_name} -limit 10]  
prompt> report_collection $sorted_cells -columns \  
    {{full_name {width 26} {fit elide_center}}} \  
    area number_of_pins} -max_rows 10
```

SEE ALSO

- `collections(2)`
- `filter_collection(2)`
- `sort_collection(2)`
- `write_collection(2)`
- `list_attributes(2)`

report_compile_options

Displays information about the **compile** command options for the design of the current instance if set; or for the current design otherwise.

SYNTAX

status **report_compile_options**
[-nosplit]

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

This command displays the **compile** command directives of the current instance or current design and all of the designs in the hierarchy. The compile directives determine how a design is optimized.

If you set **current_instance**, the report is generated for the design of that instance; otherwise the report is generated for the current design.

EXAMPLES

The following is an example of a compile options report for the design *AN2_DESIGN*:

```
prompt> report_compile_options
```

```
*****  
Report : compile_options  
Design : AN2_DESIGN  
Version: 1998.02  
Date   : Fri Dec 29 14:24:08 1997  
*****
```

Design	Compile Option	Value

AN2_DESIGN	flatten	false
	structure	true
	structure_boolean	false
	structure_timing	true

In the following example, compile options are set for the design *encoder*, and a report is generated:

prompt> **set_structure false**

prompt> **set_flatten true**

prompt> **report_compile_options**

```
*****
Report : compile_options
Design : encoder
Version: 1998.02
Date  : Fri Dec 29 14:24:08 1997
*****
```

Design	Compile Option	Value

encoder	flatten	true
	flatten_effort	low
	flatten_minimize	single
	flatten_phase	false
	structure	false
	structure_boolean	false
	structure_timing	false

In the following example, cost priority and multiple port nets options are set for the design *encoder*, and a report is generated:

prompt> **set_cost_priority -delay**

prompt> **set_fix_multiple_port_nets -feedthroughs -outputs**

prompt> **report_compile_options**

```
*****
Report : compile_options
Design : encoder
Version: 1998.02
Date  : Fri Dec 29 14:24:08 1997
*****
```

Design	Compile Option	Value

encoder	flatten	true
	flatten_effort	low
	flatten_minimize	single
	flatten_phase	false
	structure	false

cost_priority max_delay
fix_multiple_port_nets feedthroughs
 outputs

SEE ALSO

compile(2)
set_cost_priority(2)
set_fix_multiple_port_nets(2)
set_flatten(2)
set_structure(2)

report_compile_spg_mode

Displays the tool settings updated by the **set_compile_spg_mode** command.

SYNTAX

status **report_compile_spg_mode**

ARGUMENTS

This command has no arguments.

DESCRIPTION

The **report_compile_spg_mode** command displays the tool settings updated by the **set_compile_spg_mode** command. The **set_compile_spg_mode** command sets tool settings to improve timing correlation with IC Compiler II or IC Compiler.

The command is supported only in topographical mode.

EXAMPLES

The following example shows the usage of the command:

```
prompt> report_compile_spg_mode
```

SEE ALSO

[set_compile_spg_mode\(2\)](#)

report_congestion

Reports the congestion statistics.

SYNTAX

```
status report_congestion
  [-effort minimum | low | medium | high]
  [-list_cells_over_grc_violation grc_violation_threshold]
```

Data Types

grc_violation_threshold integer

ARGUMENTS

-effort minimum | low | medium | high

Specifies the global routing effort level when the tool generates a global route congestion map.

The lower effort has a faster runtime but results in a more pessimistic congestion number. The higher effort has a better congestion result but requires a longer runtime. The default is **medium**.

This option applies only when the routing stage is **global**.

-list_cells_over_grc_violation *grc_violation_threshold*

Reports the cells in the congested region when GRC violation threshold is provided. Cells are presorted to list design with most congested cells.

DESCRIPTION

Note that congestion optimization and reporting is available only with Design Compiler Graphical.

This command computes and displays congestion statistics for the current design.

If you call this command two or more consecutive times without any changes to the design, global routing is skipped for the second time and the report is based on the global routing done in the first command.

For congestion analysis purposes, the design is divided into small regions called global routing cells (GRCs). A given global routing cell is considered overcongested if the actual number of wires passing through the global routing cell is greater than the number of available tracks in the global routing cell.

By default, the available tracks in the global routing cell are automatically filled based on the layer pitch. That is, the space between adjacent tracks always equals the layer pitch, not the distance of two actual tracks. If you want to use actual tracks instead of layer-pitch-based tracks, set the **dct_enable_track_auto_fill** variable to **false**.

In the congestion report, the Overflow value is the total number of wires in the design global routing cells that do not have a corresponding track available. The Max value corresponds to the highest number of overutilized wires in a single global routing cell. The GRCs value is the total number of overcongested global routing cells in the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **report_congestion** command:

```
prompt> report_congestion
```

```
*****
Report : congestion
Design : test_design
Version: A-2007.12
Date   : Mon Oct 29 10:43:51 2007
*****
```

```
Both Dirs: Overflow = 5 Max = 1 (5 GRCs) GRCs = 5 (11.90%)
H routing: Overflow = 5 Max = 1 (5 GRCs) GRCs = 5 (11.90%)
V routing: Overflow = 0 Max = 0 (0 GRCs) GRCs = 0 (0.00%)
```

1

```
prompt> report_congestion -list_cells_over_grc_violation 30
```

```
*****
Report : congestion
Design : many_rom
Version: I-2013.12-SP1-CS2
Date   : Mon Jan 6 22:00:39 2014
*****
```

```
Both Dirs: Overflow = 590305 Max = 33 (1 GRCs) GRCs = 85938 (67.13%)
H routing: Overflow = 275009 Max = 29 (1 GRCs) GRCs = 43141 (33.70%)
V routing: Overflow = 315296 Max = 33 (1 GRCs) GRCs = 42797 (33.43%)
```

Cells in Congested Regions:

Design	Cell	Reference	Congestion
many_rom	sub_Syn1st27909	SBC5LND2V1D1	33
many_rom	sub_Syn1st28217	SBC5LND2V1D2	31

1

SEE ALSO

report_congestion_options(2)
set_congestion_options(2)

report_congestion_options

Reports congestion options in the design.

SYNTAX

```
status report_congestion_options  
[-all]  
id_list
```

Data Types

id_list list

ARGUMENTS

-all

Report all congestion options.

id_list

Reports congestion options with the specified IDs.

DESCRIPTION

The **report_congestion_options** command reports the congestion options in the design. If **-all** is used, all congestion options in the design are reported. If a list of IDs is specified, only congestion options with the specified IDs are reported. The command first reports the design-wide congestion options. It will then report the regional congestion options, if any. The regional congestion options report includes the ID, the coordinates, and the values.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example report of the congestion options:

```
prompt> report_congestion_options -all
```

```
Report congestion options:
```

```
*****
```

```
Congestion parameters for the design:
```

```
Horizontal threshold: 0.8 (default)
```

```
Vertical threshold: 0.8 (default)
```

```
Maximum utilization: 0.95 (default)
```

```
Regional congestion options:
```

```
ID 0: {0 0 20 20}, MAX_UTIL: 0.5
```

```
*****
```

SEE ALSO

`remove_congestion_options(2)`

`report_congestion(2)`

`set_congestion_options(2)`

report_constraint

Displays constraint-related information about a design.

SYNTAX

```
status report_constraint
  [-all_violators]
  [-verbose]
  [-significant_digits digits]
  [-max_area]
  [-max_delay]
  [-critical_range]
  [-min_delay]
  [-max_capacitance]
  [-min_capacitance]
  [-max_transition]
  [-max_fanout]
  [-cell_degradation]
  [-max_toggle_rate]
  [-min_porosity]
  [-max_dynamic_power]
  [-max_leakage_power]
  [-max_total_power]
  [-max_net_length]
  [-connection_class]
  [-multiport_net]
  [-nosplit]
  [-min_pulse_width]
  [-min_period]
  [-scenarios scenario_list]
  [-ignore_infeasible_paths]
```

Data Types

```
digits      integer
scenario_list list
```

ARGUMENTS

-all_violators

Displays a summary of all of the optimization and design rule constraints with violations in the current design. The **-verbose** option provides detailed information about constraint violations. Multiple violations for a given constraint are listed in order from largest to smallest violation.

-verbose

Displays more detail about constraint calculations.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported. The *digits* value must be between 0 and 13. The default is 2. This option overrides the value set by the **report_default_significant_digits** variable.

-max_area

Displays only the max_area constraint information. The default behavior (without this option and similar options) is to display all optimization and design rule constraints.

-max_delay

Displays only the max_delay and setup information.

-critical_range

Displays only the critical_range information. The critical_range is a design rule that directs the tool to optimize near-critical paths along with the most critical path.

-min_delay

Displays only the min_delay and hold information.

-max_capacitance

Displays only the max_capacitance constraint information. The max_capacitance constraint is a design rule that limits the total capacitance on a net.

-min_capacitance

Displays only the min_capacitance constraint information. The min_capacitance constraint is a design rule that ensures a minimum total capacitance on a net.

-max_transition

Displays only the max_transition constraint information. The max_transition constraint is a design rule that limits the transition time on a net. If the library uses the cmos2 delay model, this option shows the max_edge_rate information.

-max_fanout

Displays only the max_fanout constraint information. The constraint is a design rule that limits the fanout_load on a net.

-cell_degradation

Displays only the cell_degradation constraint information. The cell_degradation constraint is a design rule that limits the total capacitance on a net, with the limit depending on the transition times at the inputs of the cell.

-max_toggle_rate

Displays only the max_toggle_rate constraint information.

-min_porosity

Displays only the min_porosity constraint information. The min_porosity constraint is an optimization constraint for routability.

-max_dynamic_power

Displays only the max_dynamic_power constraint information. The default behavior (without this option and similar power-related options) is to display all types of power constraint information. Queries for power constraint information are valid only if a power-

related license is available.

-max_leakage_power

Displays only the max_leakage_power constraint information.

-max_total_power

Displays only the max_total_power constraint information.

-max_net_length

Displays only max_net_length constraint information. The max_net_length constraint is a design rule that limits the route length of a net. For more information, see the man page for the **set_max_net_length** command.

-connection_class

Displays only the connection_class constraint information. The connection_class constraint is displayed only if there is a connection_class violation. For more information, see the man page for the **set_connection_class** command.

-multiport_net

Displays only the multiport_net constraint information. This constraint specifies whether multiple output ports can be connected to a given net. For more information, see the man page for the **set_fix_multiple_port_nets** command.

-nosplit

Prevents line splitting and facilitates writing applications to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the width of the column, the next field begins on a new line, starting in the correct column.

-min_pulse_width

Displays only the min_pulse_width constraint information. The min_pulse_width constraint is a design rule that limits the minimum duration of clock pulses in the clock network.

-min_period

Displays only the minimum period constraint information. The min_period constraint is a design rule that sets a minimum period on a clock signal. The min_period check is supported only for ideal clocks.

-scenarios *scenario_list*

Reports constraints for the specified scenarios of a multi-scenario design. Each scenario is reported separately. Inactive scenarios are not reported.

If you do not use this option, the **report_constraint** command reports constraints on all active scenarios, except when you use the **-all_violators** or **-verbose** option. If you use these options but not the **-scenarios** option, the **report_constraint** command reports constraints only on the current scenario.

-ignore_infeasible_paths

Ignores all the paths flagged as infeasible during the latest compilation.

DESCRIPTION

The **report_constraint** command displays the following information for the constraints on the current design:

- Whether the constraint was violated or met
- By how much the constraint value was violated or met
- The design object that was the worst violator

The maximum delay information shows cost by path group. This includes violations of setup time on registers or ports with output delay, as well as violations of **set_max_delay** commands. The total maximum delay cost is the sum of each group's weighted cost. For details on creating path groups, refer to the **group_path** command man page. To see the current path groups in the design, use the **report_path_group** command.

The minimum delay cost includes violations of hold time on registers or ports with output delay as well as violations of **set_min_delay** commands.

In the path delay reports, if a pin drives a high-fanout net, this is indicated in the report by a # symbol between the incremental and path timing values. Creation and usage of scenarios is available with Design Compiler Graphical.

Multicorner-Multimode Support

By default, this command uses information from all active scenarios. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following example shows brief constraint information for the current design:

```
prompt> report_constraint
```

```
*****
Report : constraint
Design : counter
Version: 1998.02
Date   : Fri Dec 26 15:49:46 1997
*****
```

		Weighted	
Group (max_delay/setup)	Cost	Weight	Cost
CLK	0.00	1.00	0.00
default	0.00	1.00	0.00

max_delay/setup			0.00

		Total Neg	Critical	
Group (critical_range)	Slack	Endpoints	Cost	
CLK	0.00	0	0.00	
default	0.00	0	0.00	

critical_range			0.00	

Constraint	Cost

max_transition	0.00 (MET)

max_fanout	0.00 (MET)
max_delay/setup	0.00 (MET)
sequential_clock_pulse_width	0.00 (MET)
critical_range	0.00 (MET)
min_delay/hold	0.40 (VIOLATED)
max_leakage_power	6.00 (VIOLATED)
max_dynamic_power	14.03 (VIOLATED)
max_area	48.00 (VIOLATED)

The following example displays detailed constraint information for the current design:

prompt> **report_constraint -verbose**

Report : constraint
-verbose

Design : counter

Version: v3.1a

Date : Tue 1992

Startpoint: ffb (rising edge-triggered flip-flop clocked by CLK)

Endpoint: ffd (rising edge-triggered flip-flop clocked by CLK)

Path Group: CLK

Path Type: max

Point	Incr	Path

clock CLK (rise edge)	0.00	0.00
startpoint clock skew (ideal)	0.00	0.00
startpoint clock uncertainty	0.00	0.00
ffb/CP (FD3)	0.00	0.00 r
ffb/QN (FD3)	2.42	2.42 r
w/Z (ND4)	0.59	3.01 f
q/Z (EO)	1.13	4.14 f
j/Z (AO2)	1.08	5.22 r
ffd/D (FDS2)	0.00	5.22 r
data arrival time		5.22

clock CLK (rise edge)	10.00	10.00
endpoint clock skew (ideal)	0.00	10.00
endpoint clock uncertainty	0.00	10.00
ffd/CP (FDS2)	0.00	10.00 r
library setup time	-0.90	9.10
data required time		9.10

data required time		9.10
data arrival time		-5.22

slack (MET)		3.88

Design: counter

max_area	30.00
- Current Area	78.00

Slack -48.00 (VIOLATED)

Design: counter

max_leakage_power 70.00
- Current Leakage Power 76.00

Slack -6.00 (VIOLATED)

Design: counter

max_dynamic_power 500.00
- Current Dynamic Power 514.03

Slack -14.03 (VIOLATED)

The following example displays detailed information on only those constraints that have violations:

prompt> **report_constraint -all_violators -verbose**

Report : constraint

-all_violators

-verbose

Design : led

Version: v3.2a

Date : Tue Jan 3 13:00:45 1995

Startpoint: b (input port)

Endpoint: z5 (output port)

Path Group: default

Path Type: max

Point	Incr	Path

input external delay	0.00	0.00 r
b (in)	0.00	0.00 r
U5/Z (IV)	1.32	1.32 f
U3/Z (NR2)	3.35	4.67 r
U18/Z (AO6)	0.73	5.40 f
U22/Z (AO4)	1.42	6.82 r
z5 (out)	0.00	6.82 r
data arrival time		6.82
max_delay	6.50	6.50
output external delay	0.00	6.50
data required time		6.50

data required time		6.50
data arrival time		-6.82

slack (VIOLATED)		-0.32

Startpoint: c (input port)

Endpoint: z3 (output port)

Path Group: default

Path Type: max

Point	Incr	Path

input external delay	0.00	0.00 r
c (in)	0.00	0.00 r
U6/Z (IV)	1.34	1.34 f
U2/Z (NR2)	3.35	4.69 r
U15/Z (AO7)	0.87	5.56 f
U24/Z (AO3)	1.02	6.57 r
z3 (out)	0.00	6.57 r
data arrival time		6.57
max_delay	6.50	6.50
output external delay	0.00	6.50
data required time		6.50

data required time		6.50
data arrival time		-6.57

slack (VIOLATED)		-0.07

Net: a

max_transition	1.00
- Transition Time	1.26

Slack	-0.26 (VIOLATED)

Net: a

max_fanout	5.00
- Fanout	7.00

Slack	-2.00 (VIOLATED)

Design: led

max_area	30.00
- Current Area	36.00

Slack	-6.00 (VIOLATED)

Design: led

max_dynamic_power	1000.00
- Current Dynamic Power	1254.81

Slack	-254.81 (VIOLATED)

The following example displays the max_area, max_delay/setup, min_delay/hold, and max_leakage_power constraint information:

```
prompt> report_constraint -max_area -max_delay -min_delay \
-max_leakage_power
```

```
*****
```

```
Report : constraint
-max_area
```

-max_delay
-min_delay
-max_leakage_power

Design : led
Version: v3.2a
Date : Tue Jan 3 13:00:56 1995

		Weighted	
Group (max_delay/setup)	Cost	Weight	Cost

default	0.32	1.00	0.32

max_delay/setup			0.32
Constraint	Cost		

max_delay/setup			0.32 (VIOLATED)
max_area			6.00 (VIOLATED)

SEE ALSO

- create_clock(2)
- group_path(2)
- report_clock(2)
- report_design(2)
- report_path_group(2)
- report_timing(2)
- report_timing_requirements(2)
- report_min_pulse_width(2)
- set_critical_range(2)
- set_fix_multiple_port_nets(2)
- set_max_area(2)
- set_max_delay(2)
- set_max_dynamic_power(2)
- set_max_leakage_power(2)

report_cross_probing

Reports cross-probing data for specified cells and ports.

SYNTAX

```
status report_cross_probing  
  [-nosplit]  
  [-original_hierarchy]  
  objects
```

Data Types

objects collection

ARGUMENTS

-nosplit

Prevents line splitting in the cross-probing report. Most of the design information is listed in fixed-width columns. In the absence of this option, if the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-original_hierarchy

Changes the format of the report by adding an additional line per object to display an object's original RTL hierarchy. Sometimes a cell undergoes a series of operations, such as ungrouping, name changes, and optimization that collapses the hierarchy and renames the cell. The **-original_hierarchy** option displays the RTL hierarchical name by which the cell was originally created and helps to identify where the cell originated.

objects

Specifies the cell and port objects to show in the report.

DESCRIPTION

This command displays cross-probing information about specified cells and ports in a report format.

You must have a design that was analyzed and elaborated or synthesized using Design Compiler version I-2013.12 or later. The tool provides information about where the cell or port was created, specifying which file and line number the object came from. If the object was tool generated, the file and line number are not applicable and are represented by a hyphen (-). In this case, the report specifies whether the origin of the object comes from UPF, the DFT engine, or another source.

Objects in the design that were merged can result in an object with multiple source locations in the report. Objects with multiple source positions appear multiple times in the report, and, for clarity, the object name is indented for lines other than the first source line.

The tool reports the following information for each specified object:

Type : The type of design object, either a cell or port.

Reference : For cells, this is the library reference cell. For ports, this is either IN, OUT, or INOUT.

Filename : The base name of the file. If this information is not available, the tool displays a hyphen (-).

Line : The line number associated with the file. If this information is not available, the tool displays a hyphen (-).

Origin : The origin associated with the object. For example, this value can be rtl, datapath, upf, clock_gating, self_gating, or dft.

If this information is not available, the tool displays a hyphen (-).

Object : The object name. If an object has multiple source positions, the object name is indented for lines other than the first source line.

Object / Original Hierarchy : If the -original_hierarchy option is used, you will see this column instead of the Object column.

For each object, the first row in this column shows the object name.

For each object, the second row in this column shows the object's original hierarchy.

At the end of the report, mapping from the base name to the absolute file name is provided. You need to verify the existence of the file because the design or original source files might have been moved or deleted after the design was read in.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows a **report_cross_probing** report:

```
prompt> set objects [get_ports]
{top_i top_o}
prompt> set objects [add_to_collection $objects [get_cells -hierarchical]]
{top_i top_o mid1/bot1/c1 mid1/bot1/c2 mid1/bot2/c1 mid1/bot2/c2 mid1/foo1/c2 mid1/bot1 mid1/bot2
mid1/foo1 mid1}
prompt> report_cross_probing $objects
```

```
*****
```

Report : cross_probing

Design : top

Version: ...

Date : ...

```
*****
```

Type	Reference	Filename	Line	Origin	Object
port	IN	m.v	2	rtl	top_i
port	OUT	m.v	3	rtl	top_o
cell	IVA	m.v	21	rtl	mid1/bot1/c1
cell	IVA	m.v	22	rtl	mid1/bot1/c2
cell	IVA	m.v	21	rtl	mid1/bot2/c1
cell	IVA	m.v	22	rtl	mid1/bot2/c2
cell	IVA	m.v	28	rtl	mid1/foo1/c2

cell	bot	m.v	12	rtl	mid1/bot1
cell	bot	m.v	13	rtl	mid1/bot2
cell	myfoo	m.v	14	rtl	mid1/foo1
cell	mid	m.v	4	rtl	mid1

Filename	Full Pathname

m.v	/usr/joe/design/m.v

The following example shows a report with the objects' original hierarchy information:

```
prompt> report_cross_probing [get_cells U1*] --original_hierarchy
*****
Report   : cross_probing_files
Design  : data_switch_512
Version : ...
Date    : ...
*****
```

Type	Reference	Filename	Line	Origin	Object/Original Hierarchy

cell	HS65_STF_AOI	test1.v	21536	rtl	U129
		I_CORE/I_SUB1			
cell	HS65_STF_AND2	test2.v	21548	rtl	U104
		I_CORE/I_SUB2			

Filename	Full Pathname

test1.v	/usr/joe/rtl/test1.v
test2.v	/usr/joe/rtl/test2.v

SEE ALSO

get_cross_probing_info(2)
cross_probing_filter(2)

report_cross_probing_files

Shows a list of cross-probing files and their status.

SYNTAX

```
status report_cross_probing_files  
[-errors_only]
```

ARGUMENTS

-errors_only

Reports files whose status are not OK.

DESCRIPTION

This command prints out a list of the cross-probing files that are stored in the database and the status of each file. The file status can be one of the following:

Status:

OK - No issues found

Missing - File is missing

No Permissions - User has no read permissions

Modified - File contents have been modified

If the status of a file is marked as Modified, the file content has been modified since the file was read. The Missing status might also indicate that the path exists, but it is a directory.

You can use this command to determine if the set of cross-probing file paths stored in the database are valid before using the GUI for cross-referencing. You can also list which RTL files need to be copied to the destination site if the design is being moved to a new location.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of the **report_cross_probing_files** command:

```
prompt> report_cross_probing_files
```

```
*****  
Report : cross_probing_files  
Design : top  
Version: K-2015.06  
Date   : Wed Dec 10 14:09:15 2014  
*****
```

Status:

OK - No issues found

Missing - File is missing

No Permissions - User has no read permissions

Modified - File contents have been modified

Filename	Status

/common/rtl/bot.v	OK
/usr/site/joe/designs/mid.v	OK
/usr/site/joe/designs/top.v	OK

SEE ALSO

update_cross_probing_files(2)

report_crpr

Reports the clock reconvergence pessimism calculated between specified register clock pins or ports.

SYNTAX

```
status report_crpr  
  -from from_latch_clock_pin  
  -to to_latch_clock_pin  
  [-from_clock from_clock]  
  [-to_clock to_clock]  
  [-setup | -hold]  
  [-significant_digits digits]
```

Data Types

```
from_latch_clock_pin  string  
to_latch_clock_pin    string  
from_clock            string  
to_clock              string  
digits                integer
```

ARGUMENTS

-from *from_latch_clock_pin*

Specifies the clock pin of the launching sequential device or clock-gating check to be reported. A constrained input port may also be used.

-to *to_latch_clock_pin*

Specifies the clock pin of the capturing sequential device or clock-gating check to be reported. A constrained output port may also be used.

-from_clock *from_clock*

Specifies the clock that fans out to the launching sequential device.

-to_clock *to_clock*

Specifies the clock that fans out to the capturing sequential device.

-setup

Reports the clock reconvergence pessimism for a setup data path. The **-setup** and **-hold** options are mutually exclusive. If neither option is specified, **-setup** is assumed.

-hold

Reports the clock reconvergence pessimism for a hold data path. The **-setup** and **-hold** options are mutually exclusive. If neither option is specified, **-setup** is assumed.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0 through 13. If this option is not specified, the number of significant digits is determined by the **report_default_significant_digits** variable, which has a default value of 2.

DESCRIPTION

This command reports the clock reconvergence pessimism calculated between specified register clock pins or ports. The command displays the following information about the calculation of the clock reconvergence pessimism between the two specified sequential elements:

- The name of the common pin in the clock network
- The clock edges (rising or falling) that propagate through the common point to the launching and capturing registers
- The value of the **timing_crpr_threshold_ps** variable
- A tabulation of the arrival times for both clock edges at the common point and their associated clock reconvergence pessimism (crp_rise and crp_fall)
- The clock reconvergence pessimism used along with the reasoning behind the selection of crp_rise or crp_fall used for that report
- The potential range of accuracy of the clock reconvergence pessimism value that will appear in the corresponding timing report is displayed. The range of accuracy depends upon the value of the **timing_crpr_threshold_ps** variable.

If the **timing_remove_clock_reconvergence_pessimism** variable is set to **false**, clock reconvergence pessimism removal is inactive and no report can be generated. To determine the current value of the variable, type the following command:

```
prompt> printvar timing_remove_clock_reconvergence_pessimism
```

The **-from_clock** and **-to_clock** options allow you to generate a report for only the specified clocks. By default, a report is issued for all clocks that fanout to each sequential device.

Two clock reconvergence pessimism values are calculated for each potential common point in the design:

- The crp_rise value is calculated from rising arrival times at the common point.
- The crp_fall value is calculated from falling arrival times at the common point.

The value of clock reconvergence pessimism used in the report depends upon what clock edges propagate through the common point to the clock pins of the launching and capturing devices. For example, if a rising edge through the common point triggers the launching device and also triggers the capturing device, then a rising clock reconvergence pessimism value (crp_rise) is used. Similar reasoning applies for a falling edge propagating through the common point and subsequently launching and capturing data leading to a falling clock reconvergence pessimism value being used (crp_fall). If a rising edge through the common point launches the data and a falling edge through the common point captures it, then a mismatch in sense occurs.

If a mismatch occurs and the **timing_clock_reconvergence_pessimism** variable is set to **normal**, then the minimum of crp_rise and crp_fall is used. If a mismatch occurs and the variable is set to **same_transition**, then the clock reconvergence pessimism is reported as 0. This selection process is reported in the selection details section of the report. To determine the current value of the **timing_clock_reconvergence_pessimism** variable, type the following command:

```
prompt> printvar timing_clock_reconvergence_pessimism
```

When the capturing sequential device is a level-sensitive latch, 2 clock reconvergence pessimism values are reported. The first value corresponds to the opening edge of the latch and appears in the timing report. The second value corresponds to the closing edge of the device. The selection details section also reports the decision-making process behind both clock reconvergence pessimism values. In this case, since the opening and closing clock edges at the latch will always be different, there will always be a mismatch in clock edges for one of them.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example reports the clock reconvergence pessimism calculated between the *ffa* and *ffd* sequential elements for a setup data path:

```
prompt> report_crpr -from ffa/CP -to ffd/CP -setup
```

```
*****
Report : CRP Calculation
Design : counter
Version: 2004.12
Date   : Mon Nov 1 15:44:20 2004
*****
```

```
Startpoint: ffa (rising edge-triggered flip-flop clocked by CLK)
Endpoint:   ffd (rising edge-triggered flip-flop clocked by CLK)
```

```
Common Point: CLK <inside>
Common Clock: CLK
Launching edge at common point: RISING
Capturing edge at common point: RISING
CRPR threshold: 0.02
```

Arrival Times	Early	Late	CRP
Rise	3.00	19.00	16.00
Fall	5.00	23.00	18.00

Selection Details

```
Edge Match:      Match, using rise CRP
```

```
clock reconvergence pessimism          16.00
```

```
Range of accuracy of CRP in report_timing, due to value
of timing_crpr_threshold_ps:  15.98 <= CRP <= 16.00
```

1

The following example displays a report where the launching device is a flip-flop and the capturing device is a latch. In this case, the

mismatch in clock edges occurs for the clock reconvergence pessimism corresponding to the closing edge at the latch.

prompt> **report_crpr -from reg1/CP -to lat2/G**

Report : CRP Calculation

Design : borrow

Version: 2004.12

Date : Mon Nov 1 15:44:20 2004

Startpoint: reg1 (rising edge-triggered flip-flop clocked by clk)

Endpoint: lat2 (positive level-sensitive latch clocked by clk)

Common Point: clk_buf/Z

Common Clock: clk

Launching edge at common point: RISING

Latch open edge at common point: RISING

CRPR threshold: 0.01

Arrival Times	Early	Late	CRP
Rise	2.1229	2.6529	0.5300
Fall	2.0868	2.7868	0.7000

Selection Details

Edge Match (opening): Match, using rise CRP

Edge Match (closing): Mismatch, using min(rise CRP, fall CRP)

clock reconvergence pessimism (open edge)	0.5300
clock reconvergence pessimism (close edge)	0.5300

Range of accuracy of CRP in report_timing, due to value
of timing_crpr_threshold_ps: 0.5200 <= CRP <= 0.5300

1

SEE ALSO

timing_clock_reconvergence_pessimism(3)

timing_crpr_threshold_ps(3)

timing_remove_clock_reconvergence_pessimism(3)

report_datapath_gating

Reports the status of datapath gating in the current design.

SYNTAX

```
status report_datapath_gating  
  [-instances]  
  [-nosplit]  
  [-ungated]  
  [-gated]
```

ARGUMENTS

-instances

Specifies that a breakdown per instance of the number of gated and ungated operators should be reported.

-nosplit

Prevents line-splitting and facilitates writing applications to extract information from the report output. Most design information is listed in fixed-width columns. If information for a field exceeds the column width, the next field begins on a new line, starting in the correct column.

-ungated

Print out all the ungated operators.

-gated

Print out all the gated operators with their optimization modes.

DESCRIPTION

The **report_datapath_gating** command reports the status of datapath gating in the current design. The report is divided into two sections, one section reports datapath gating breakdown per instance and the other section reports a gating breakdown global summary. The global summary is generated when the **-instances** option is not used. The ungated cell number reported in the instance report and ungated report might underestimate the number of ungated cells. Some cells that are ungrouped due to small bit-width might not be reported.

EXAMPLES

The following example is a sample output of the **report_datapath_gating** command output without options.

The output is limited to a summary of the datapath gating performed in the entire design.

```
prompt>report_datapath_gating
```

```
*****
Report : datapath_gating
Design : sample
Version: C-2009.06
Date   : Fri Jan 23 02:25:20 2009
*****
```

Library(s) Used:

power_lib (File: /root/libraries/power_lib.db)

Datapath Gating Summary

Gating Style	AND
Number of Operators	2 (100.00%)
Number of Gated operators	1 (50.00%)
Number of Ungated operators	1 (50.00%)

The next example shows the datapath gating breakdown per instance. Note that the global summary is reported as well.

```
prompt>report_datapath_gating -instances
```

```
*****
Report : datapath_gating
        -instances
Design : sample
Version: C-2009.06
Date   : Fri Jan 23 02:25:20 2009
*****
```

Library(s) Used:

power_lib (File: /root/libraries/power_lib.db)

Breakdown per Instance

Gated	Style	Ungated	Parent Instance
1	AND	0	sample
0		1	U1

Datapath Gating Summary

Gating Style	AND
Number of Operators	2 (100.00%)
Number of Gated operators	1 (50.00%)
Number of Ungated operators	1 (50.00%)

To determine the ungated operators, use the **-ungated** option with the **report_datapath_gating** command, as shown in the following example. Note that the global summary is reported as well.

```
prompt>report_datapath_gating -ungated
```

```
*****
Report : datapath_gating
Design : top
Version: D-2010.03-BETA4
Date   : Fri Jan 15 20:02:04 2010
*****
```

Library(s) Used:

power_lib (File: /root/libraries/power_lib.db)

Datapath Gating Summary

Gating Style	AND
Number of Operators	12 (100.00%)
Number of Gated operators	1 (8.33%)
Number of Ungated operators	11 (91.67%)

Ungated Cell Report

Ungated Cell	Reason
Design top:	
lte_x_244_2	Small DW design
lt_x_815_1	Small DW design
lt_x_815_5	Small DW design
lte_x_244_4	Small DW design
lte_x_244_3	Small DW design
DP_OP_416J1_296_9011	Small DW design
add_x_244_1	Small DW design
DP_OP_417J1_297_6567	No gating opportunity
add_x_1078_1	Timing violation or no power gain
DP_OP_418J1_298_9691	No gating opportunity
DP_OP_424_300_8749	Timing violation or no power gain

```

-----
Total number of ungated cell:      11
No gating opportunity:           2 (18.18%)
Timing violation or no power gain: 2 (18.18%)
Small DW design:                 7 (63.64%)

```

To determine the gated operators, use the **-gated** option with the **report_datapath_gating** command, as shown in the following example. Note that the global summary is reported as well.

```
prompt>report_datapath_gating -gated
```

```

*****
Report : datapath_gating
Design : test
Version: I-2013.12-SP1-CS1
Date  : Tue Dec 10 18:47:55 2013
*****

```

Library(s) Used:

```
power_lib (File: /root/libraries/power_lib.db)
```

Datapath Gating Summary

Gating Style	AND
Number of Operators	2 (100.00%)
Number of Gated operators	1 (50.00%)
Number of Ungated operators	1 (50.00%)

Gated Cell Report

Gated Cell	Opt. Mode

Design test:

mult_x_1	Power

```

-----
Total number of gated cell:      1
Gated in 'power' mode:          1 (100.00%)

```

SEE ALSO

```

set_datapath_gating_options(2)
power_enable_datapath_gating(3)

```

report_delay_calculation

Displays the actual calculation of a timing arc delay value for a cell or net.

SYNTAX

```
status report_delay_calculation
  -min
  -max
  -from from_pin
  -to to_pin
  [-nosplit]
  [-crosstalk]
  [-from_rise_transition from_rise_value]
  [-from_fall_transition from_fall_value]
  [-derate]
```

Data Types

```
from_pin      string
to_pin        string
from_rise_value float
from_fall_value float
```

ARGUMENTS

-min

Specifies that the report must show how a minimum delay value is computed. If you do not specify this option, the report shows how a maximum delay value is computed.

-max

Specifies that the report must show how a maximum delay value is computed. This is the default behavior. This option cannot be used with the **-min** option.

-from *from_pin*

Specifies the starting point of a timing arc within a design. This option must be used along with **-to** *to_pin*. When this option is specified, you cannot specify a collection of multiple pins. For a cell timing arc, the pins must represent a common leaf cell's input and output pins, which have a timing arc specified between them in the library. For a net timing arc, the pins must be a driver and a load on a common net. The specified pins must be associated with library cell instances connected by the net. Port names are allowed in place of a pin name for net arcs. A hierarchical pin is not a valid pin for this option.

-to *to_pin*

Specifies the ending point of a timing arc within a design. This option must be used along with **-from** *from_pin*. When this option is specified, you cannot specify a collection of multiple pins. For a cell timing arc, the pins must represent a common leaf cell's input and output pins, which have a timing arc specified between them in the library. For a net timing arc, the pins must be a driver and a load on a common net. The specified pins must be associated with library cell instances connected by the net. Port names are allowed in place of a pin name for net arcs. A hierarchical pin is not a valid pin for this option.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the delay data is listed in fixed-width columns. If the text for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-crosstalk

Reports the crosstalk information for a net arc. The arc is specified by *from_pin* and *to_pin*.

-from_rise_transition from_rise_value

Specifies the from rise transition value used by the delay calculation. The specified value is in main library units.

-from_fall_transition from_fall_value

Specifies the from fall transition value used by the delay calculation. The specified value is in main library units.

-derate

Reports derating components and derated delay.

DESCRIPTION

This command provides detailed timing calculation information about the specified cell or net timing arc. You can use this information for debugging or verifying timing data in a technology library. Before using this command to display details of a cell timing arc, read the technology library file into the system using the **read_lib** command. The **read_lib** command automatically compiles the library and enables the **report_delay_calculation** command for cell timing arcs. Otherwise, the built-in security mechanism terminates the command when issued for a cell timing arc. This restriction does not apply to net timing arcs.

Both operating conditions and wire-load models are taken into account when making delay calculations. Timing ranges are not taken into account because they typically apply to an entire path.

The **-crosstalk** option on net arc reports the aggressor and victim information for both rise and fall. The option prints coupling capacitance, the driving library cell, and the clocks reaching the net (both victim and aggressor). The aggressors have "Switching Bump," which is used for prioritizing the aggressors and also have aggressor attributes (active or screened).

The **-derate** option reports the derating components and derated delay about the specified cell or net timing arc. It displays AOCV/POCV derating information if AOCV/POCV is applied on the specified arc.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following examples show reports generated using **report_delay_calculation**:

prompt> **report_delay_calculation -from BLK1/A -to BLK1/Z**

Report : delay_calculation

Design : led

Version: v3.1

Date : Tue Apr 7 16:52:43 1992

From: BLK1/A

To: BLK1/Z

arc type: cell

arc sense: inverting

Input net transition times: Dt_rise = 0, Dt_fall = 0

Rise Delay computation:

rise_intrinsic	0 +
rise_slope * Dt_fall	0 * 0 +
rise_resistance * (pin_cap + wire_cap) / driver_count	
0.2 * (0 + 0.53) / 1	
rise_transition_delay	0.106

Total	0.106

Fall Delay computation:

fall_intrinsic	0 +
fall_slope * Dt_rise	0 * 0 +
fall_resistance * (pin_cap + wire_cap) / driver_count	
0.15 * (0 + 0.53) / 1	
fall_transition_delay	0.0795

Total	0.0795

prompt> **report_delay_calculation -from BLK1/Z -to BLK2/A**

Report : delay_calculation

Design : led

Version: v3.1

Date : Tue Apr 7 16:28:07 1992

From: BLK1/Z

To: BLK2/A

arc type: net

Operating Conditions: BASIC_WORST Library: basic

Wire Load Model Mode: top

Design	Wire Loading Model	Library

RDC_GENERIC	BASIC_ONE	basic

Balanced case tree

equation : $(r\_wire/load\_count) * (c\_pins + c\_wire/load\_count)$

$(0.53 / 1) * (1 + 0.53 / 1)$

delay rise, fall : 0.608175 , 0.608175

prompt> **report_delay_calculation **

**-from exetop0/e_dptop0/e_flag0/I27/Y **

-to exetop0/e_dptop0/e_flag0/U1/A -crosstalk

Report : delay_calculation

Design : Xtalk_test

Version: A-2007.12

Date : Tue Sep 25 23:18:25 2007

From pin: exetop0/e_dptop0/e_flag0/I27/Y

To pin: exetop0/e_dptop0/e_flag0/U1/A

* Some/all delay information is back-annotated.

Operating Conditions: slow Library: slow

Annotated max rise net delta delay: 0.011865 arc delay: 0.014991

Annotated max fall net delta delay: 0.006210 arc delay: 0.009337

Annotated max rise net delta transition: 0.025998 pin transition: 0.247500

Annotated max fall net delta transition: 0.013779 pin transition: 0.240000

Reporting for Crosstalk:

Victim net name: exetop0/e_dptop0/e_flag0/N194

Number of aggressors: 5

Number of effective (non-filtered) aggressors: 5

Victim driver rail voltage(VDD): 1.620000

Attributes:

A - aggressor is Active

S - aggressor is screened

Victim is rising:

Victim Net	Coupling Cap	Driver Lib Cell	Clocks

exetop0/e_dptop0/e_flag0/N194	0.013922	MX4X4	CK108M

Aggressor Net	Coupling Cap	Driver Lib Cell	Clocks	Attributes (ratio of VDD)	Switching Bump

exetop0/e_rndm0/n35	0.004685	INVX4	CK108M	A	0.023494
exetop0/e_rndm0/n18454	0.001537	SEDFFX4	CK108M	A	0.018957
exetop0/e_rndm0/n10	0.003462	BUFX2	CK108M	A	0.017911

```

exetop0/e_rndm0/n19178
      0.001858  INVX2  CK108M  S      -
exetop0/s_AEI2_0_
      0.002379  SEDFFX4  CK108M  S      -

```

Victim is falling:

Victim Net	Coupling Cap	Driver Lib Cell	Clocks
exetop0/e_dptop0/e_flag0/N194	0.013922	MX4X4	CK108M

Aggressor Net	Coupling Cap	Driver Lib Cell	Clocks	Attributes (ratio of VDD)	Switching Bump
exetop0/e_rndm0/n18454	0.001537	SEDFFX4	CK108M	A	0.016719
exetop0/e_rndm0/n35	0.004685	INVX4	CK108M	A	0.014618
exetop0/e_rndm0/n10	0.003462	BUFX2	CK108M	S	-
exetop0/e_rndm0/n19178	0.001858	INVX2	CK108M	S	-
exetop0/s_AEI2_0_					

prompt> **report_delay_calculation -from BLK1/Z -to BLK2/A -derate**

Report: delay_calculation

Design: top

Version: I-2013.12-SP

Date: Thu Jun 19 20:15:50 2014

From: BLK1/Z

To: BLK2/A

arc type: net

Arnoldi-based Delay Calculation:

RC network on pin 'BLK1/Z' :

Number of elements = 18 Capacitors + 17 Resistors

Total capacitance = 0.064393 pF

Total capacitance = 0.064393 (in main library unit)

Total resistance = 0.578063 Kohm

	Rise	Fall
Net delay	= 0.000436	0.000436 (in main library unit)
Transition time	= 1.284843	0.733576 (in main library unit)
From_pin transition time	= 1.284839	0.733568 (in main library unit)
To_pin transition time	= 1.284843	0.733576 (in main library unit)
Net slew degradation	= 0.000005	0.000008 (in main library unit)

	Rise	Fall
Net derate	= 1.200000	1.000000
Net delay derated	= 0.000524	0.000436 (in main library unit)

SEE ALSO

- report_lib(2)
- report_timing(2)
- rc_driver_model_mode(3)
- rc_receiver_model_mode(3)

report_delay_estimation_options

Reports the parameters that influence delay estimation. This command is supported only in topographical mode.

SYNTAX

status **report_delay_estimation_options**

ARGUMENTS

The **report_delay_estimation_options** command has no arguments.

DESCRIPTION

The **report_delay_estimation_options** command reports the parameters previously set with the **set_delay_estimation_options** command. These parameters influence preroute delay estimation used during placement and routing.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following commands set and report the preroute delay estimation options:

```
prompt>set_delay_estimation_options \  
-min_unit_horizontal_capacitance 0.0015 \  
-min_unit_vertical_capacitance 0.0010 \  
-min_unit_horizontal_resistance 0.15 \  
-min_unit_vertical_resistance 0.10 \  
-density_outside_block 0.2
```

```
prompt> report_delay_estimation_options
```

...

Delay estimation options for the design:

Minimum horizontal unit capacitance: 0.0015
Minimum vertical unit capacitance: 0.001
Minimum horizontal unit resistance: 0.15
Minimum vertical unit resistance: 0.1
Density outside block: 0.2

SEE ALSO

`set_delay_estimation_options(2)`

report_design

Displays attributes of the current design.

SYNTAX

```
status report_design
      [-nosplit]
      [-physical]
```

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Design information is listed in fixed-width columns. If information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-physical

This option is ignored by Design Compiler.

DESCRIPTION

The **report_design** command lists information about the attributes of the design.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following is an example of a design report:

```
prompt> report_design
```

```
*****
```

```
Report : design
```

```
Design : my_design
```


Version: F-2011.09
Date : Tue Jun 7 15:16:17 2011

Design allows ideal nets on clock nets.

Library(s) Used:

sc_max (File: /usr/lib/ref/db/sc_max.db)
io_max (File: /usr/lib/ref/db/io_max.db)

Local Link Library:

{sc_max.db}

Flip-Flop Types:

No flip-flop types specified.

Latch Types:

No latch types specified.

Operating Conditions:

Operating Condition Name : sc_max
Library : sc_max
Process : 1.20
Temperature : 125.00
Voltage : 1.08
Interconnect Model : worst_case_tree

Min Operating Conditions:

Operating Condition Name : sc_min
Library : sc_min
Process : 0.80
Temperature : -40.00
Voltage : 1.32
Interconnect Model : best_case_tree

Wire Loading Model:

Selected from the default.

Name : ForQA
Location : sc_max
Resistance : 0.00038
Capacitance : 9.6e-05
Area : 0.01
Slope : 15.9634
Fanout Length Points Average Cap Std Deviation

1 5.88
2 13.03
...

19 199.57
20 215.53

Wire Loading Model Mode: enclosed.

Timing Ranges:

No timing ranges specified.

Pin Input Delays:

Pin	Input Delay				Clock
	Min		Max		
	Rise	Fall	Rise	Fall	
U1/A	4.50	4.50	4.50	4.50	--

Pin Output Delays:

None specified.

Disabled Timing Arcs:

No arcs disabled.

Required Licenses:

None Required

Design Parameters:

width => 16

1

SEE ALSO

report_clock(2)
report_internal_loads(2)
report_port(2)

report_design_lib

Lists the design units contained in the specified libraries.

SYNTAX

```
status report_design_lib  
  [-libraries]  
  [-designs]  
  [-architectures]  
  [-packages]  
  [library_list]
```

Data Types

library_list list

ARGUMENTS

-libraries

Indicates that libraries, and not their contents, are to be listed.

-designs

Indicates that designs in libraries (Verilog modules, VHDL entities, or configurations) are to be listed.

-architectures

Indicates that designs in libraries, plus their architectures, are to be listed.

-packages

Indicates that packages in specified libraries are to be listed.

library_list

Specifies a list of libraries whose contents are to be displayed. If no *library_list* is specified, all libraries are displayed.

DESCRIPTION

The **report_design_lib** command lists the contents of specified libraries.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **report_design_lib** to list the contents of the current design libraries:

```
prompt> report_design_lib

*****

Report : design libraries
Version: v3.0
Date   : Tue Oct 1 14:44:53 1991
*****

Contents of current design libraries
  DEFAULT (/users/Jane_Doe/design/test/work)
  WORK (/users/Jane_Doe/design/test/work)
    package : pack1
    entity   : add
    architecture : d fast(add)
    architecture : m d small(add)
    entity : p mult
    architecture : m verilog(mult)
  TYPES_LIB (/users/Jane_Doe/design/report/types_lib)
    package : n top_pack
  FUNCS_LIB (/users/Jane_Doe/design/report/funcs_lib)
    package : s pack
  SYNOPSIS (/users/Jane_Doe/design/top/dc/libraries/syn/packages)
    package : ATTRIBUTES
    package : TYPES
    package : synopsys

p --- This design has parameters.
m --- This architecture is the most recently analyzed.
n --- Can't find the source for this file.
s --- This file is out of date with respect to its source.
d --- This file is out of date with respect to its depends.
```

SEE ALSO

```
analyze(2)
define_design_lib(2)
elaborate(2)
read_file(2)
```

report_design_mismatch

Reports design mismatches that were circumvented to allow linking the design.

SYNTAX

```
status report_design_mismatch
  [-summary]
  [-class all | netlist | library]
```

ARGUMENTS

-summary

Summarizes the report.

-class all | netlist | library

Specifies whether to print netlist, library, or all mismatches.

DESCRIPTION

The **report_design_mismatch** command reports netlist mismatches that were circumvented (by modifying the netlist) to allow linking the design. Circumventing design mismatches prevents formal verification.

EXAMPLES

The following example shows an output of the **report_design_mismatch** command.

```
prompt> report_design_mismatch

*****
Report : design mismatches
Design : TCS03
Version: 2010.12
Date   : Wed Mar 19 14:57:08 2010
*****
```

Design Mismatches

Logical

Physical

Missing physical pin	(LINK-905)	2	1
Missing physical view for logical part (LINK-908)			

Total netlist elements	0	3	

1

mb.SH "SEE ALSO"

link_allow_design_mismatch(3)

report_dft_clock_controller

Reports the on-chip clocking controller specification for the current design.

SYNTAX

```
status report_dft_clock_controller  
[-view existing_dft | spec]
```

ARGUMENTS

-view existing_dft | spec

Indicates the view to which the command applies. Valid views are **existing_dft** and **spec**.

Setting the view to **spec** indicates that the command applies to DFT-inserted clock controllers. This is the default.

Setting the view to **existing_dft** indicates that the command refers to user-defined clock controllers.

DESCRIPTION

This command displays the current on-chip clocking controller specification applied to the current design.

Use the **set_dft_clock_controller** command to specify an on-chip clocking controller specification. Use the **reset_dft_clock_controller** command to reset the current on-chip clocking controller specification.

For more information, see the man page for the **set_dft_clock_controller** command.

EXAMPLES

The following command reports the clock_controller controller specification for the current design.

```
prompt> report_dft_clock_controller -view spec
```

```
*****
```

```
Report : Clock controller
```

```
Design : top
```

```
Version: G-2012.06-SP3
```

```
Date   : Tue Nov 13 12:46:05 2012
```

=====

TEST MODE: all_dft

VIEW : Specification

=====

Cell name: CORE

Design: snps_clk_mux

Chain count: 1

Cycle count: 2

PLL clock: UPLL/CLKO

ATE clock: ATECLK

1

SEE ALSO

reset_dft_clock_controller(2)

set_dft_clock_controller(2)

report_dft_clock_gating_configuration

Displays the options specified by the **set_dft_clock_gating_configuration** command.

SYNTAX

status **report_dft_clock_gating_configuration**

ARGUMENTS

This command takes no arguments.

DESCRIPTION

This command displays the options that were specified with the **set_dft_clock_gating_configuration** command on the current design.

EXAMPLES

The following example shows the report generated by the **report_dft_clock_gating_configuration** command:

```
prompt> set_dft_clock_gating_configuration -exclude_elements \  
    {U_block_a/cg1 U_block_b/cg1}  
Accepted clock gating configuration specification  
1  
prompt> set_dft_clock_gating_configuration \  
    -dont_connect_cgs_of I_ADD/add_out_reg[0]  
Accepted clock gating configuration specification  
1  
prompt> report_dft_clock_gating_configuration
```

```
*****  
Report : DFT Clock gating configuration  
Design : top  
Version: D-2010.03-SP2  
Date   : Tue May 18 15:33:21 2010  
*****
```

Excluded Clock Gating Cells: 2

U_block_a/cg1

U_block_b/cg1

Registers to exclude Clock Gating Cells: 1

I_ADD/add_out_reg[0]

1

SEE ALSO

set_dft_clock_gating_configuration(2)

reset_dft_clock_gating_configuration(2)

report_dft_clock_gating_pin

Displays the specification specified by the **set_dft_clock_gating_pin** command.

SYNTAX

integer **report_dft_clock_gating_pin**

ARGUMENTS

This command takes no arguments.

DESCRIPTION

This command displays the options that were specified with the **set_dft_clock_gating_pin** command on the current design.

When the reported test pin of a clock-gating cell is active low and controlled by a ScanEnable or TestMode signal, the control signal is reported as ScanEnable_inverted or TestMode_inverted, respectively.

EXAMPLES

The following example shows the report generated by the **report_dft_clock_gating_pin** command. Consider a set of **set_dft_clock_gating_pin** commands as follows:

```
set_dft_clock_gating_pin {sub1/clk_gate_out1_reg} -pin_name TE \  
-control_signal ScanEnable -active_state 1
```

```
set_dft_clock_gating_pin {sub2/clk_gate_out1_reg} -pin_name TE \  
-control_signal ScanEnable -active_state 0
```

```
set_dft_clock_gating_pin {sub3/clk_gate_out1_reg} -pin_name TE \  
-control_signal TestMode -active_state 1
```

```
set_dft_clock_gating_pin {sub4/clk_gate_out1_reg} -pin_name TE \  
-control_signal TestMode -active_state 0
```

The output of the **report_dft_clock_gating_pin** command for these commands is:

```
prompt> report_dft_clock_gating_pin
*****
```

```
Report : DFT Clock gating pin configuration
```

```
Design : top
```

```
Version: D-2010.03-SP4-CS1-0802
```

```
Date   : Wed Jul 28 17:22:54 2010
```

```
*****
```

```
User Identified Clock Gating Cells: 4
```

```
-----
Control Signal   : Pin       : Cell
-----
```

```
ScanEnable       : TE        : sub1/clk_gate_out1_reg
```

```
ScanEnable_inverted : TE      : sub2/clk_gate_out1_reg
```

```
TestMode         : TE        : sub3/clk_gate_out1_reg
```

```
TestMode_inverted : TE       : sub4/clk_gate_out1_reg
```

```
1
```

SEE ALSO

```
set_dft_clock_gating_pin(2)
```

```
remove_dft_clock_gating_pin(2)
```

report_dft_configuration

Displays the options specified by the **set_dft_configuration** command.

SYNTAX

status **report_dft_configuration**

ARGUMENTS

The **report_dft_configuration** command has no arguments.

DESCRIPTION

This command displays options specified by the **set_dft_configuration** command on the current design.

The command shows which DFT clients are enabled or disabled.

EXAMPLES

The following is an example of the report generated by the **report_dft_configuration** command:

```
prompt> report_dft_configuration
```

```
*****  
Report : DFT configuration  
Design : top  
Version: M-2016.12  
Date   : Mon Oct 31 07:11:46 2016  
*****
```

DFT Structures	Status
-----	-----
Scan:	Enable
Fix Sets:	Disable
Fix Resets:	Disable
Fix Clocks:	Disable

Fix Busses:	Enable
Fix Bdirectional Ports:	Enable
Fix X Propagation:	Disable
Test Points:	Disable
Testability:	Enable
Logic BIST:	Disable
Wrapper:	Disable
Boundary scan:	Disable
Scan Compression:	Disable
Streaming Compression:	Disable
Core Integration:	Disable
Power Control:	Disable
Pipeline Scan Data:	Disable
Clock Controller:	Disable
ConnectClockGating:	Enable
Mode Decoding Style:	Binary
IEEE 1500 control:	Disable

SEE ALSO

report_dft_configuration(2)
report_dft_signal(2)
report_scan_configuration(2)
report_scan_path(2)
set_dft_configuration(2)
set_dft_signal(2)
set_scan_configuration(2)
set_scan_path(2)

report_dft_connect

Reports the existing DFT connectivity specifications.

SYNTAX

status **report_dft_connect**

ARGUMENTS

The **report_dft_connect** command has no arguments.

DESCRIPTION

The **report_dft_connect** command reports the current DFT connectivity associations. The command reports associations specified by the **set_dft_connect** command.

EXAMPLES

The following command generates a report of current DFT connectivity specifications:

```
prompt> report_dft_connect
```

SEE ALSO

[remove_dft_connect\(2\)](#)
[set_dft_connect\(2\)](#)
[set_dft_signal\(2\)](#)

report_dft_design

Reports all user-specified DFT designs.

SYNTAX

```
status report_dft_design
  [-all]
  [-type design_type_name]
  [-design_name design_name]
```

Data Types

```
design_type_name  string
design_name      string
```

DESCRIPTION

The **report_dft_design** command reports all user-specified DFT designs in the following format:

```
Design Name    Design Type
-----
<name1>       <type1>
...
-----
```

EXAMPLES

The following example reports all user-specified DFT designs:

```
prompt> define_dft_design -design_name my_des \
  -type WC_D1 -interface {shift_clk capture_clk h \
  capture_en capture_en h shift_en shift_dr h \
  cti si h cto so h cfi data_in h cfo data_out h}
1

prompt> define_dft_design -design my_des1 -type BC_4 \
  -interface {capture_clk cap_clk h capture_en cen h \
  shift_dr shift h si ti h so to h data_in di h data_out do h}
1

prompt> define_dft_design -design my_des2 -type BC_4 \
```



```
-interface {capture_clk cap_clk h capture_en cen h \  
  shift_dr shift h si ti h so to h data_in di h data_out do h}  
1
```

prompt> **report_dft_design**

Design Name	Design Type

my_des	WC_D1

my_des1	BC_4

my_des2	BC_4

SEE ALSO

define_dft_design(2)

insert_dft(2)

preview_dft(2)

remove_dft_design(2)

report_dft_drc_rules

Reports DFT DRC specifications that affect how certain DRC rule violations affect DFT insertion.

SYNTAX

```
status report_dft_drc_rules  
  [-violation drc_list]  
  [-cell cell_list]
```

Data Types

```
drc_list    string  
cell_list   string
```

ARGUMENTS

-violation *drc_list*

Restricts the report to the specified DRC violations.

-cell *cell_list*

Restricts the report to the specified cells.

DESCRIPTION

The **report_dft_drc_rules** command reports modifications to how certain DRC rule violations affect DFT insertion, as specified by the **set_dft_drc_rules** command. The list of violation IDs supported by this command is:

- TEST-504 (warning) Cell %s has constant 0 value.
- TEST-505 (warning) Cell %s has constant 1 value.
- D17 (warning) D17 Clock input I of <type> S couldn't capture data.

By default, the command reports all DFT DRC specifications applied by the **set_dft_drc_rules** command. You can use the **-violation** and **-cell** options to restrict the scope of the report.

Use the **set_dft_drc_rules** command to apply DFT DRC rule specifications. Use the **reset_dft_drc_rules** command to reset all DFT DRC rule specifications.

EXAMPLES

The following example applies some DFT DRC specifications to the design:

```
prompt> set_dft_drc_rules -warning {TEST-504}
```

```
prompt> set_dft_drc_rules -warning {TEST-505} -cell {BLK1 BLK2}
```

```
prompt> set_dft_drc_rules -ignore {TEST-505} -cell {USPAREGATES}
```

The following command reports all DFT DRC specifications:

```
prompt> report_dft_drc_rules
```

Violation Name	Default Action	Specified Action	Range/ Cell list
TEST-504	omit	allow	all cells
TEST-505	omit	allow	BLK1
	omit	allow	BLK2
	omit	ignore	USPAREGATES

The following command restricts the report to a single rule type:

```
prompt> report_dft_drc_rules -violation {TEST-504}
```

Violation Name	Default Action	Specified Action	Range/ Cell list
TEST-504	omit	allow	all cells

The following command restricts the report to specific cells:

```
prompt> report_dft_drc_rules -cell {BLK1 BLK2}
```

Violation Name	Default Action	Specified Action	Range/ Cell list
TEST-504	omit	allow	all cells
TEST-505	omit	allow	BLK1
TEST-505	omit	allow	BLK2

SEE ALSO

dft_drc(2)
 insert_dft(2)
 preview_dft(2)
 reset_dft_drc_rules(2)
 set_dft_drc_rules(2)

report_dft_equivalent_signals

Reports all of the equivalent DFT signals specified with **set_dft_equivalent_signals** command.

SYNTAX

status **report_dft_equivalent_signals**

ARGUMENTS

The **report_dft_equivalent_signals** command has no arguments.

DESCRIPTION

This command reports the equivalent set of signals specified using the **set_dft_equivalent_signals** command.

EXAMPLES

The following example reports all equivalent scan signals:

```
prompt> report_dft_equivalent_signals
```

SEE ALSO

insert_dft(2)
preview_dft(2)
remove_dft_equivalent_signals(2)
set_dft_equivalent_signals(2)
set_dft_signal(2)

report_dft_hierarchical_pins

Reports the ports and hierarchical pins created during DFT insertion.

SYNTAX

```
status report_dft_hierarchical_pins
  [-added_by_dft true | false]
  [-ports_only false | true]
  [-summary_only false | true]
```

ARGUMENTS

-added_by_dft true | false

When set to **true**, ports and pins created during DFT insertion are reported. When set to **false**, reports uniquely ports that existed prior to DFT insertion. The default is **true**.

-ports_only false | true

Controls whether to report only ports, or ports and hierarchical pins. The default is **false**.

-summary_only false | true

Specifies whether to report a summary instead of the entire object table. The default is **false**.

DESCRIPTION

This command displays a report of ports and/or hierarchical pins created during DFT insertion.

EXAMPLES

The following example shows reports generated by **report_dft_hierarchical_pins**:

```
prompt> report_dft_hierarchical_pins

*****
Report : Boundary Pins added during DFT
Design : cntl
```

Version: N-2017.09-SP4
 Date : Wed Apr 11 10:40:43 2018

Amount of Boundary Pins added during DFT : 6
 Amount of Pins added during DFT : 0
 Amount of Ports added during DFT : 6

```
-----
#    Type    Pin Name
-----
1    port    data_source
2    port    test_mode
3    port    test_si1
4    port    test_si2
5    port    test_so2
6    port    test_se
```

1

prompt> **report_dft_hierarchical_pins -added_by_dft false**

 Report : Boundary Pins added before DFT
 Design : cntl
 Version: N-2017.09-SP4
 Date : Wed Apr 11 10:40:57 2018

Amount of Boundary Pins added before DFT : 7
 Amount of Pins added before DFT : 0
 Amount of Ports added before DFT : 7

```
-----
#    Type    Pin Name
-----
1    port    cond1
2    port    cond2
3    port    cond3
4    port    enab
5    port    clk
6    port    reset
7    port    ccond
```

1

SEE ALSO

get_dft_hierarchical_pins(2)
 current_design(2)
 insert_dft(2)

report_dft_insertion_configuration

Displays options set by the **set_dft_insertion_configuration** command.

SYNTAX

status **report_dft_insertion_configuration**

ARGUMENTS

The **report_dft_insertion_configuration** command has no arguments.

DESCRIPTION

This command displays options set by the **set_dft_insertion_configuration** command on the current design.

EXAMPLES

The following is an example of the **report_dft_insertion_configuration** command:

```
prompt> report_dft_insertion_configuration
```

```
*****  
Report : DFT insertion configuration  
Design : A  
Version: 2003.12  
Date   : Wed Aug 20 17:37:20 2003  
*****
```

Options	Status
-----	-----
Map_effort	Medium
Preserve_design_name	False
Route_scan_enable	False
Route_scan_clock	False
Route_scan_serial	False
Synthesis_optimization	All

Unscan	False
1	

SEE ALSO

report_dft_configuration(2)
report_dft_signal(2)
report_scan_path(2)
report_scan_configuration(2)
set_dft_configuration(2)
set_dft_signal(2)
set_scan_configuration(2)
set_scan_path(2)

report_dft_location

Reports the DFT hierarchy location specifications for the current design.

SYNTAX

```
status report_dft_location
      [-all]
```

ARGUMENTS

-all

Reports all DFT hierarchy location specifications.

DESCRIPTION

The **report_dft_location** command is used to report the DFT hierarchy location specification for the current design.

EXAMPLES

The following example reports the default DFT hierarchy locations for various logic types for the design named *des_unit*:

```
prompt> current_design des_unit
```

```
Current design is 'des_unit'.
```

```
{des_unit}
```

```
prompt> report_dft_location
```

Design Name	DFT PARTITION NAME	DFT TYPE	DFT Hierarchy Location
des_unit	default_partition	BSR	<top>
des_unit	default_partition	CODEC	<top>
des_unit	default_partition	LOCKUP_LATCH	<local>
des_unit	default_partition	XOR_SELECT	<top>
des_unit	default_partition	RETIMING_FLOP	<local>
des_unit	default_partition	TAP	<top>
des_unit	default_partition	TCM	<top>
des_unit	default_partition	PIPELINE_SE_LOGIC	<top>
des_unit	default_partition	PIPELINE_SI_LOGIC	<top>

```

des_unit      default_partition  PLL          <local>
des_unit      default_partition  REC_MUX      <local>
des_unit      default_partition  SERIAL_CNTRL <top>
des_unit      default_partition  SERIAL_REG   <top>
des_unit      default_partition  WRAPPER      <top>
des_unit      default_partition  IEEE_1500    <top>
1

```

The following example reports user specified DFT hierarchy locations for the design named *des_unit*:

```

prompt> current_design des_unit
Current design is 'des_unit'.
{des_unit}
prompt> set_dft_location -exclude {REC_MUX LOCKUP_LATCH BSR TAP} UTEST_LOGIC
Warning: Specified hierarchy name 'UTEST_LOGIC' doesn't exists in the current design 'des_unit'. (UIT-1112)
Accepted DFT location specification.
1
prompt> set_dft_location -include {TCM} UTCM
Warning: Specified hierarchy name 'UTCM' doesn't exists in the current design 'des_unit'. (UIT-1112)
Accepted DFT location specification.
1
prompt> report_dft_location
Design Name      DFT PARTITION NAME    DFT TYPE      DFT Hierarchy Location
=====
des_unit         default_partition      BSR           <top>
des_unit         default_partition      CODEC          UTEST_LOGIC
des_unit         default_partition      LOCKUP_LATCH   <local>
des_unit         default_partition      XOR_SELECT     <top>
des_unit         default_partition      RETIMING_FLOP  <local>
des_unit         default_partition      TAP            <top>
des_unit         default_partition      TCM            UTCM
des_unit         default_partition      PIPELINE_SE_LOGIC  UTEST_LOGIC
des_unit         default_partition      PIPELINE_SI_LOGIC  UTEST_LOGIC
des_unit         default_partition      PLL            UTEST_LOGIC
des_unit         default_partition      REC_MUX        <local>
des_unit         default_partition      SERIAL_CNTRL    UTEST_LOGIC
des_unit         default_partition      SERIAL_REG      UTEST_LOGIC
des_unit         default_partition      WRAPPER        UTEST_LOGIC
des_unit         default_partition      IEEE_1500      UTEST_LOGIC
1

```

SEE ALSO

```

insert_dft(2)
remove_dft_location(2)
set_dft_location(2)

```

report_dft_partition

Displays design partition information attached to a design.

SYNTAX

status **report_dft_partition**

ARGUMENTS

The **report_dft_partition** command has no arguments.

DESCRIPTION

The **report_dft_partition** command displays design partition information about a design.

EXAMPLES

The following example shows a design partition report for the *CORE* design:

```
prompt> report_dft_partition
```

```
*****  
Report : DFT PARTITION Definition  
Design : CORE  
Version: B-2008.09  
Date   : Fri Jun 13 16:36:29 2008  
*****
```

```
Cells or Designs defined in Partition 'part1':
```

```
U1  
U2
```

```
Cells or Designs defined in Partition 'part2':
```

```
U3  
U4  
U5
```

SEE ALSO

current_dft_partition(2)
define_dft_partition(2)
insert_dft(2)
preview_dft(2)
remove_dft_partition(2)
set_dft_signal(2)
set_scan_compression_configuration(2)
set_scan_configuration(2)
set_scan_path(2)

report_dft_power_control

Reports the power controller block specification for the current design.

SYNTAX

status **report_dft_power_control**

ARGUMENTS

The **report_dft_power_control** command has no arguments.

DESCRIPTION

This command displays the current power controller block specification applied to the current design.

Use the **set_dft_power_control** command to specify a power controller block specification. Use the **remove_dft_power_control** command to remove the current power controller block specification.

For more information, see the man page for the **set_dft_power_control** command.

EXAMPLES

The following command reports the power controller block specification for the current design.

```
prompt> report_dft_power_control  
Power Controller Instance Name: 'U_PWC'  
1
```

SEE ALSO

[remove_dft_power_control\(2\)](#)
[set_dft_power_control\(2\)](#)

report_dft_signal

Displays options specified by the **set_dft_signal** command.

SYNTAX

```
status report_dft_signal
  [-view spec | existing_dft]
  [-test_mode mode_name_list | all]
  [-port list_of_port_names]
  [-type signal_type]
```

Data Types

```
mode_name_list    list
list_of_port_names list
signal_type      string
```

ARGUMENTS

-view spec | existing_dft

Specifies the view to report. Valid values are **spec** and **existing_dft**. If not specified, the **spec** view is the default.

-test_mode *mode_name_list* | all

Specifies the test mode(s) to report. If not specified, the tool reports on the current test mode, as displayed by the **current_test_mode** command. If no test modes were created, the tool reports on the default test mode. If **all** is specified, the tool displays the configurations of all test modes.

-port *list_of_port_names*

Specifies the name of a port or a list of ports on which to report. If no ports are specified, the tool reports on all defined ports.

-type *signal_type*

Specifies the signal type on which to report.

DESCRIPTION

This command displays options set by the **set_dft_signal** command on the current design.

The command reports on signal types, active states, hook-up pins, and timing of the ports.

EXAMPLES

The following command specifies a report on the test mode named *mymodeA*:

```
prompt> report_dft_signal -test_mode mymodeA
```

```
*****
```

```
Report : DFT signals
```

```
Design : A
```

```
Version: 2003.12
```

```
Date : Thu Aug 14 10:46:08 2003
```

```
*****
```

```
=====
TEST MODE: mymodeA
```

```
VIEW : Specification
```

```
=====
Port      SignalType  Active Hookup Timing
-----
u5/A      ScanEnable    1    -
SE_A      ScanEnable    1    u5/A
SE_ALL    ScanEnable    -    -
SI        ScanDataIn  -    -    Delay 5
```

The following command specifies a report on a list of test modes:

```
prompt> report_dft_signal -test_mode {mymodeA mymodeB}
```

```
*****
```

```
Report : DFT signals
```

```
Design : A
```

```
Version: 2003.12
```

```
Date : Thu Aug 14 10:46:08 2003
```

```
*****
```

```
=====
TEST MODE: mymodeA
```

```
VIEW : Specification
```

```
=====
Port      SignalType  Active Hookup Timing
-----
u5/A      ScanEnable    -    -
SE_A      ScanEnable    -    u5/A
SE_ALL    ScanEnable    -    -
```

```
=====
TEST MODE: mymodeB
```

```
VIEW : Specification
```

```
=====
Port      SignalType  Active Hookup Timing
-----
SE_B      ScanEnable    -    -
u5/A      ScanEnable    -    -
```

SE_A	ScanEnable	-	u5/A
SE_ALL	ScanEnable	-	-

SEE ALSO

current_test_mode(2)
report_dft_configuration(2)
report_dft_signal(2)
report_scan_configuration(2)
report_scan_path(2)
set_dft_configuration(2)
set_dft_signal(2)
set_scan_configuration(2)
set_scan_path(2)

report_direct_power_rail_tie

Reports all library pins on which the **direct_power_rail_tie** attribute is set to **true**.

SYNTAX

status **report_direct_power_rail_tie**

ARGUMENTS

The **report_direct_power_rail_tie** command has no arguments.

DESCRIPTION

This command reports all library pins on which the **direct_power_rail_tie** attribute is set to **true**.

If the value of the **direct_power_rail_tie** attribute on a library pin is **true**, then all pins in the design for which this is the library pin, it won't connect it to a tieoff cell for connecting it to constant signal. Instead, it will keep them connecting to generic constant signals, so that during power routing these pins can be connected directly to power rails.

To remove the **direct_power_rail_tie** attribute, use the **remove_attribute** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command reports all library pins that have the **direct_power_rail_tie** attribute set to **true**:

```
prompt> report_direct_power_rail_tie target_lib/leaf_cell/GND  
target_lib/leaf_cell/GND
```

SEE ALSO

set_direct_power_rail_tie(2)

report_disable_timing

Reports disabled timing arcs in the current design.

SYNTAX

```
string report_disable_timing
[-nosplit]
```

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. By default, most design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line starting in the correct column.

DESCRIPTION

The **report_disable_timing** command reports disabled timing arcs in the current design. Timing arcs can be disabled in three ways. The first way is by using the **set_disable_timing** command. The second and third ways are both through automatically disabled arcs by the synthesis timing engine. This automatic disabling occurs in order to break timing loops or when propagating constants in the design.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example disables the timing arc of a cell named *U1/U2* from pin *A* to pin *Z*:

```
prompt> set_disable_timing {U1/U2} -from A -to Z
```

```
prompt> report_disable_timing
```

```
*****
```

```
Report : disable_timing
```

Design : middle

Flags : c case-analysis
C Conditional arc
l loop breaking
u user-defined
L stored loop breaking

Cell or Port	From	To	Flag
U1/U2	A	Z	u
U1/U2	B	Z	L

The following example disabled the timing arc of a cell named *ff1* from pin *CP* to pin *TE* and *TI* as a result of a case-analysis constant propagation to pin *TE* of cell *ff1*. Note that the L flag appears only when such arcs exist.

prompt> **set_case_analysis 0 {ff1/TE}**

prompt> **report_disable_timing**

Report : disable_timing
Design : middle

Flags : c case-analysis
C Conditional arc
l loop breaking
u user-defined

Cell or Port	From	To	Flag
ff4	CP	TI	c
ff4	CP	TE	c

SEE ALSO

set_case_analysis(2)
set_disable_timing(2)

report_dont_touch

Reports dont_touch cells or nets in the current design along with their dont_touch types.

SYNTAX

```
status report_dont_touch  
  [objects | -class class_name]  
  [-nosplit]
```

Data Types

objects collection
class_name string

ARGUMENTS

objects

Reports the specified dont_touch cells or nets.

The **-class** and *objects* arguments are mutually exclusive; you can use only one option.

If neither option is specified, the **report_dont_touch** command reports all cell and net dont_touch objects.

-class *class_name*

Reports dont_touch objects for the specific class along with their dont_touch types. The *class_name* value can be **cell** or **net**.

The **-class** and *objects* arguments are mutually exclusive; you can use only one option.

If neither option is specified, the **report_dont_touch** command reports all cell and net dont_touch objects.

-nosplit

Prevents line splitting. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_dont_touch** command displays information about the dont_touch cells and nets in the current design.

If the current instance is set, the report is generated for the design of that instance. Otherwise, the report is generated for the current design. Objects that do not have the **dont_touch** attribute are ignored.

Use the **-class** option to report only dont_touch cells or dont_touch nets. You can also specify a collection of cells or nets to report.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of a dont_touch report with the **-class** and **-nosplit** options:

```
prompt>report_dont_touch -class cell -nosplit
*****
Report : dont_touch
Design : des_chip
Version: E-2010.12-SP2
Date   : Tue Feb  8 19:40:39 2011
*****

Description for Cell dont_touch Types:
inh_ref      - Cell inherits dont_touch from reference design or library cell
ph_fixed_placement - Cell with fixed placement

Object          Class    Types
-----
headerfooter5_HDRDID1BWPHVT_R5_C0      cell    ph_fixed_placement
headerfooter6_HDRDID1BWPHVT_R6_C0      cell    ph_fixed_placement
u_des_soft_macro/u_8/u_mem              cell    inh_ref
-----

Total 3 dont_touch cells
1
```

SEE ALSO

```
list_dont_touch_types(2)
get_dont_touch_cells(2)
get_dont_touch_nets(2)
set_dont_touch(2)
set_dont_touch_network(2)
```

report_dp_smartgen_options

Displays datapath strategies available to the current design.

SYNTAX

status **report_dp_smartgen_options**

ARGUMENTS

The **report_dp_smartgen_options** command has no arguments.

DESCRIPTION

This command lists the enabled and disabled datapath strategies available to the current design. These strategies provide control over some of the optimizations done by datapath generators.

EXAMPLES

The following is an example of a report generated by **report_dp_smartgen_options**:

```
prompt> report_dp_smartgen_options
```

```
*****  
Datapath smart generation options...  
*****
```

```
Smart generation:      enabled  
Default values:       auto
```

SEE ALSO

```
set_dp_smartgen_options(2)
```

report_extraction_options

Reports the options that influence postroute extraction.

SYNTAX

```
status report_extraction_options
  [-scenarios scenario_list]
  [-all]
```

Data Types

scenario_list list

ARGUMENTS

-scenarios *scenario_list*

Specifies the scenarios for which to report the extraction options.

The options for each scenario are reported separately. If you specify a nonexistent scenario, the tool just skips that scenario.

By default, only the options for the current scenario are reported.

-all

Reports the setting for all extraction options, whether the it is explicitly set or not. If you do not specify the this option, the command reports only those options explicitly set with the **set_extraction_options** command.

DESCRIPTION

The **report_extraction_options** command reports the parameters that influence the postroute extraction.

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following example runs the **report_extraction_options** command:

```
prompt> set_extraction_options \  
-max_process_scale 1.4 -min_process_scale 0.6
```

```
prompt> report_extraction_options
```

```
*****
```

```
Report : extraction options
```

```
Design : module_a
```

```
Version: B-2008.09
```

```
Date   : Fri Jul 18 12:17:20 2008
```

```
*****
```

```
Max process scaling factor : 1.4
```

```
Min process scaling factor : 0.6
```

```
1
```

SEE ALSO

set_extraction_options(2)

report_fsm

Displays state-machine attributes and information for the design of the current instance or for the current design.

SYNTAX

```
status report_fsm
[-nosplit]
```

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

This command displays state-machine attributes and information for the design of the current instance or for the current design. If the current instance has been set, the report is generated for the design of that instance. Otherwise, the report is generated for the current design.

EXAMPLES

The following is an example of a state-machine report:

```
prompt> report_fsm

*****
Report : fsm
Design : BUS_ARBITRATOR
Version: v2.0
Date   : Fri Mar 20 14:11:13 1991
*****
Clock      : CLK      Sense: rising_edge
Asynchronous Reset: Unspecified
```

Encoding Bit Length: 3
Encoding style : auto

State Vector: { FF2 FF1 FF0 }

State Encodings and Order:

Grant_A : 001
Wait_A : 011
Timeout_A1 : 111
Grant_B : 010
Wait_B : 110
Timeout_B1 : 101

Preserved States: Grant_A

Merged States: None

SEE ALSO

set_fsm_encoding(2)
set_fsm_encoding_style(2)
set_fsm_minimize(2)
set_fsm_order(2)
set_fsm_preserve_state(2)
set_fsm_state_vector(2)

report_gui_stroke_bindings

Print a report on the dictionaries and the stroke-command bindings they contain..

SYNTAX

```
report_gui_stroke_bindings
[-dictionary dictionary_name]
```

ARGUMENTS

-dictionary *dictionary_name*

Specify which dictionary should have its bindings reported. If not specified then all dictionaries will be reported.

DESCRIPTION

This command takes the data returned by `get_gui_stroke_bindings` and formats it into a report that is human-readable. The report prints out the stroke-to-command bindings for all dictionaries.

This command is defined as a part of the `strokes Tcl` package, and the command is automatically imported into the global namespace when the package is loaded.

EXAMPLES

```
ca_shell> report_gui_stroke_bindings
```

Dictionary: Graphics

stroke	cmdType	cmd
-----	-----	---
5	builtin	Pan_Center_Rect
852	builtin	Zoom_Full
258	builtin	Zoom_Full
159	builtin	Zoom_Out_Rect
357	builtin	Zoom_Out_Rect
753	builtin	Zoom_In_Rect
951	builtin	Zoom_In_Rect

```
654      builtin  Pan_Center_Rect
456      builtin  Pan_Center_Rect
123698741 builtin  Zoom_In_Rect
147896321 builtin  Zoom_In_Rect
s:14789  tcl_cmd   myTclCmd
14789    tcl_cmd   myTclCmd %rect
```

SEE ALSO

```
set_gui_stroke_preferences(2)
set_gui_stroke_binding(2)
report_gui_stroke_builtins(2)
```

report_gui_stroke_builtins

Print a report on the non-Tcl commands available for stroke bindings..

SYNTAX

```
report_gui_stroke_builtinss  
[-dictionary dictionary_name]
```

ARGUMENTS

-dictionary *dictionary_name*

Specify which dictionary should have its builtins reported. If not specified then only the global builtins will be printed.

DESCRIPTION

This command takes the data returned by `get_gui_stroke_bindings` and formats it into a report that is human-readable. The report prints out the non-Tcl commands that can be used in stroke bindings.

This command is defined as a part of the `strokes Tcl` package, and the command is automatically imported into the global namespace when the package is loaded.

EXAMPLES

```
ca_shell> report_gui_stroke_builtins
```

Built-In Commands:

```
name  
----  
Zoom_In_Rect  
Zoom_Out  
Pan_Center_Rect  
Zoom_Out_Rect  
Zoom_Full  
Zoom_In
```

SEE ALSO

set_gui_stroke_preferences(2)
set_gui_stroke_binding(2)
report_gui_stroke_bindings(2)

report_hierarchy

Displays the reference hierarchy of the current instance or the current design.

SYNTAX

```
status report_hierarchy  
  [-nosplit]  
  [-full]  
  [-noleaf]
```

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-full

Displays the full hierarchy. By default, components of submodules in multiple locations in a hierarchy are listed only once. An ellipsis (...) indicates the contents of a previously-displayed module.

-noleaf

Indicates that the leaf library cells are to be excluded from the reference hierarchy report.

DESCRIPTION

This command displays the indented reference hierarchy of the current instance or the current design. If the current instance has been set, the report is generated for the design of that instance. Otherwise, the report is generated for the current design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of a hierarchy report:

```
prompt> report_hierarchy

*****
Report : hierarchy
Design : CONTROL
Version: v2.0
Date   : Fri Mar 20 14:09:24 1991
*****
```

```
CONTROL
CTLX
  AO1      tech_lib
  AO2      tech_lib
  AO3      tech_lib
  AO4      tech_lib
  AO6      tech_lib
  AO7      tech_lib
  EON1     tech_lib
  INVA     tech_lib
  NAND2    tech_lib
  NAND3    tech_lib
  NAND4    tech_lib
  NR2      tech_lib
  NR3      tech_lib
  OR2      tech_lib
  OR3      tech_lib
INV        tech_lib
JPMP
MX1P      tech_lib
NAND2     tech_lib
NAND3     tech_lib
```

SEE ALSO

```
group(2)
report_cell(2)
report_design(2)
report_lib(2)
report_reference(2)
ungroup(2)
```

report_host_options

Prints a report of multi-CPU processing options as defined by the **set_host_options** command.

SYNTAX

status **report_host_options**

ARGUMENTS

The **report_host_options** command has no arguments.

DESCRIPTION

By default, the **report_host_options** command prints a report of the multi-CPU processing options defined by the **set_host_options** command used by IC Compiler.

EXAMPLES

The following is an example of the **report_host_options** command:

```
prompt> report_host_options
```

SEE ALSO

[remove_host_options\(2\)](#)
[set_host_options\(2\)](#)

report_icc2_options

Reports the options used to invoke the IC Compiler II session from within Design Compiler Graphical or DC Explorer physical mode.

SYNTAX

```
int report_icc2_options  
    [-verbose]  
    [-check]
```

ARGUMENTS

-verbose

Prints out verbose messages, if any.

DESCRIPTION

The **report_icc2_options** command reports the options used to launch the IC Compiler II session from within Design Compiler Graphical or DC Explorer physical mode. Use the **set_icc2_options** command to specify the options used to invoke the IC Compiler II session.

EXAMPLES

The following example reports the options that were set using the **set_icc2_options** command:

```
prompt> report_icc2_options
```

```
*****  
report_icc2_options  
*****
```

Options specified for IC Compiler II DP:

Work Directory : dcg_icc2_link

Keep Work Directory : Yes

IC Compiler II Executable : /your_location/icc2_shell

IC Compiler II Library :

/LIBS/lib1.ndm

```
/LIBS/lib2.ndm
IC Compiler II golden UPF files : NIL.
IC Compiler II Technology file : NIL.
1
```

SEE ALSO

```
compile_ultra(2)
start_icc2(2)
set_icc2_options(2)
```

report_icc_dp_options

Reports the options used to invoke the floorplan exploration session from Design Compiler Graphical. If you do not specify the options using the **set_icc_dp_options** command, **report_icc_dp_options** reports the default settings.

SYNTAX

```
status report_icc_dp_options  
  [-check]  
  [-verbose]
```

ARGUMENTS

-check

Performs the startup checks and issues errors, if any.

-verbose

Prints verbose messages, if any.

DESCRIPTION

This command reports the options used to launch the floorplan exploration session from Design Compiler Graphical. Use the **set_icc_dp_options** command to specify the options used to invoke floorplan exploration. If you do not specify the options using the **set_icc_dp_options** command, **report_icc_dp_options** reports the default settings. After using **set_icc_dp_options** and **report_icc_dp_options**, use the **start_icc_dp** command to begin floorplan exploration.

EXAMPLES

The following example reports the options and performs the startup checks:

```
prompt> report_icc_dp_options -check
```

SEE ALSO

compile_ultra(2)
set_icc_dp_options(2)
start_icc_dp(2)

report_ideal_network

Displays information about ports, pins, nets, and cells on ideal networks in the current design.

SYNTAX

```
status report_ideal_network  
  [-net]  
  [-cell]  
  [-load_pin]  
  [-timing]  
  [object_list]
```

Data Types

object_list list

ARGUMENTS

-net

Lists all nets in the specified ideal networks. By default, nets are displayed for all ideal networks.

-cell

Displays all cells in the specified ideal networks. By default, cells are displayed for all ideal networks.

-load_pin

Displays load pins (boundary pins) of the specified ideal networks, along with any ideal timing set on them using the **set_ideal_latency** and **set_ideal_transition** commands. By default, load pins are displayed for all ideal networks.

-timing

Displays all internal pins (for example, non-source and non-boundary pins) in the specified ideal networks that have ideal timing set on them using the **set_ideal_latency** and **set_ideal_transition** commands. By default, internal pins with timing are displayed for all ideal networks.

object_list

Defines a list of source ports or source pins of the specified ideal networks to be displayed. By default, the source pins and source ports for all ideal networks in the current design are displayed.

DESCRIPTION

This command displays information about the ideal networks in the current design. If no arguments are specified, all ideal network sources in the current design are displayed. If you specify a list of source pins or ports, the command displays information about the ideal networks at these objects. By default, source pins, internal pins with ideal timing, load (boundary) pins, nets, and cells are displayed. You can force a subset of this information to be displayed using the options listed above.

The "Latency" and "Transition" columns show the minimum and maximum values set for both rising and falling edges. You set these values with the **set_ideal_latency** and **set_ideal_transition** commands.

Ideal networks are an extension of ideal nets that incorporate automatic propagation of the **ideal** attribute. You specify only the source ports or pins of the network; all the nets, cells, and pins on the transitive fanin of these objects are treated as ideal by the **compile** command. The **ideal_network** attribute is automatically spread by the tool, and respread as needed during **compile** optimizations.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows an ideal network report:

```
prompt> set_ideal_network {reset1 reset2}
```

```
prompt> set_ideal_latency 5.2 reset1
```

```
prompt> set_ideal_transition 0.6 reset1
```

```
prompt> report_ideal_network
```

```
*****
```

```
Report : ideal_network
```

```
Design : TEST
```

```
Version: 2001.08
```

```
Date : Mon May 7 10:36:21 2001
```

```
*****
```

Source ports and pins	Latency				Transition			
	Rise		Fall		Rise		Fall	
	min	max	min	max	min	max	min	max
top/reset1	5.20	5.20	5.20	5.20	0.60	0.60	0.60	0.60
top/reset2	--	--	--	--	--	--	--	--

The following example shows an ideal network report for the network starting at the pin named *source1*:

```
prompt> report_ideal_network source1
```

```
*****
```

```

Report : ideal_network
Design : idn_test
Version: 2001.08
Date  : Wed May 9 17:09:02 2001
*****

```

Source ports and pins	Latency				Transition			
	Rise		Fall		Rise		Fall	
	min	max	min	max	min	max	min	max
idn_test/source1	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10

Boundary pins	Latency				Transition			
	Rise		Fall		Rise		Fall	
	min	max	min	max	min	max	min	max
ff/TE	--	--	--	--	--	--	--	--
U1/B	--	--	--	--	--	--	--	--
ff/CD	--	--						

Nets

```

a
data
rst1
n7
n8

```

Cells

```

C11
U9
U3
U10

```

The following example reports the cells and nets on the ideal network sourced at pins named *source1* and *source2*:

```
prompt> report_ideal_network -cell -net {source1 source2}
```

```

*****
Report : ideal_network
Design : idn_test
Version: 2001.08
Date  : Wed May 9 17:09:02 2001
*****

```

Source ports and pins	Latency				Transition			
	Rise		Fall		Rise		Fall	
	min	max	min	max	min	max	min	max
idn_test/source1	7.10	7.50	6.20	6.20	0.20	0.30	0.20	0.40
idn_test/source2	--	--	--	--	--	--	--	--

Nets

```

source1
source2

```

data
rst1
n7

Cells

C11
U9
U3
U8
C15
U11

SEE ALSO

current_design(2)
remove_ideal_network(2)
report_cell(2)
report_constraint(2)
report_design(2)
report_net(2)
set_ideal_latency(2)
set_ideal_network(2)
set_ideal_transition(2)

report_ieee_1500_configuration

Displays the options specified by the **set_ieee_1500_configuration** command.

SYNTAX

status **report_ieee_1500_configuration**

ARGUMENTS

The **report_ieee_1500_configuration** command has no arguments.

DESCRIPTION

This command displays options applied by the **set_ieee_1500_configuration** command to the current design.

EXAMPLES

The following is an example of the report generated by the **report_ieee_1500_configuration** command:

```
prompt> report_ieee_1500_configuration
```

```
*****
Report : IEEE 1500 Configuration
Design : chip
Version: I-2013.12
Date   : Wed Dec 18 00:52:00 2013
*****
```

```
=====
TEST MODE: all_dft
VIEW      : Specification
=====
```

IEEE 1500 Structures	Status
WIR Width:	1
CDR Width:	1

Existing WIR: FALSE
CDR Bits:
CDR Opcodes:

SEE ALSO

report_ieee_1500_configuration(2)
set_ieee_1500_configuration(2)

report_ignored_layers

Reports the routing layers that are ignored during congestion analysis and RC estimation. This command is supported only in topographical mode.

SYNTAX

status **report_ignored_layers**

ARGUMENTS

The **report_ignored_layers** command has no arguments.

DESCRIPTION

This command reports the routing layers that are ignored during congestion analysis and RC estimation.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the **report_ignored_layers** command:

```
prompt> report_ignored_layers
```

SEE ALSO

remove_ignored_layers(2)
report_lib(2)
set_ignored_layers(2)

report_inbound_cell

Displays information about inbound cells in the current instance or current design.

SYNTAX

```
status report_inbound_cell
      [-summary]
      [-hierarchical]
      [cells]
```

Data Types

cells collection

ARGUMENTS

-summary

Displays only the summary part of the report. Other arguments will be ignored.

-hierarchical

Report inbound cells in all submodules of the specified object. If this option is not specified, do not report inbound cells in submodules.

cells

Specifies the inbound cells and instances to show in the report. If not specified, all inbound cells in the current instance are listed.

DESCRIPTION

This command displays statistics about inbound cells in the current design. It also displays information about all inbound cells in the current instance or current design. If *current_instance* is set, cells in the design of that instance will be listed; otherwise cells in the current design will be listed.

For clarification on cell information shown in the report, visit the man page of `report_cell` command.

EXAMPLES

The following is an example of an inbound cell report:

prompt> **report_inbound_cell**

```
*****
Report : inbound_cell
Design : CONTROL
Version: N-2017.09-SP2
Date   : Wed Nov 22 02:41:41 2017
*****
```

Inbound Cell Design Summary

Inbound max cell percentage	10.00%	
Number of inbound cells used	14	
Percentage of inbound cells used	9.35%	

Inbound cells in current instance/design

Attributes:

n - noncombinational
d - dont_touch

Cell	Reference	Library	Area	Width	Attributes
U151	BUFFPPNN	tech_lib	0.37	1	
U191	BUFFPPNN	tech_lib	0.37	1	
U592	INVPPNN	tech_lib	0.22	1	
U318	IND4PPNN	tech_lib	0.74	1	
U138	NR2PPNN	tech_lib	0.29	1	d
U54	ND2PPNN	tech_lib	0.22	1	n
Total 6 cells		2.21			

SEE ALSO

report_cell(2)
current_design(2)
current_instance(2)
compile_inbound_cell_optimization(3)
compile_inbound_max_cell_percentage(3)

report_interclock_relation

Displays common multiple-clock periods between clocks of different periods or edge information for paths launched and captured by two different clocks. For multicorner-multimode designs, this command reports the interclock relations in the current design.

SYNTAX

```
status report_interclock_relation
  [-from from_clock_names
  | -rise_from from_clock_names
  | -fall_from from_clock_names] [-to to_clock_names]
  | -rise_to to_clock_names
  | -fall_to to_clock_names]
  [-nosplit]
  [-significant_digits digits]
  [-edge]
  [-setup]
  [-hold]
```

Data Types

```
from_clock_names  string
to_clock_names    string
digits            integer
```

ARGUMENTS

-from *from_clock_names*

Restricts reporting to only the paths that are launched by these clocks.

-rise_from *from_clock_names*

Restricts reporting to only the paths that are launched by the rising edges of these clocks.

-fall_from *from_clock_names*

Restricts reporting to only the paths that are launched by the falling edges of these clocks.

-to *to_clock_names*

Restricts reporting to only the paths that are captured by these clocks.

-rise_to *to_clock_names*

Restricts reporting to only the paths that are captured by the rising edges of these clocks.

-fall_to_clock_names

Restricts reporting to only the paths that are captured by the falling edges of these clocks.

-nosplit

Prevents line splitting in the report, which makes it easier to extract information from the report for other applications.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are 0 through 13. The default is 2. Using this option overrides the value set by the **report_default_significant_digits** variable.

-edge

Displays the launch and capture edge information for a paths driven by two different clocks.

-setup

Displays the edge information for only the setup paths. This option can be used only with the **-edge** option.

-hold

Displays the edge information for only the hold paths. This option can be used only with the **-edge** option.

DESCRIPTION

This command displays information about the timing relationships between different clocks that control data transfers between registers in a timing path.

In cases where the clock periods are not the same, the least common multiple of the clock periods becomes the common clock period. Each clock is expanded by repeating the original clock waveform until the common clock period is reached. The new longer waveform becomes the clock used for interclock timing. Depending on the clock periods, the clock expansion can fail if the common clock period is too large.

By default, each line of this report shows a pair of clocks, one that launches a timing path and one that captures the data at the end of the path. The line shows the period of each clock and the common multiple of each period. If no common period is reached between those two clocks, a dash appears where the common period is normally displayed.

If you use the **-edge** option, the command reports the launch and capture edge information for the path driven by two different clocks. Each line of the report shows a pair of clocks, one that launches a timing path and one that captures the data at the end of the path. It displays the period of each clock and the launch and capture edge information, including the edge type (rising or falling), for both the setup and hold paths, having the most restrictive timing requirement.

If you use the **-from** option, the report is restricted to paths launched by the specified clock.

If you use the **-to** option, the report is restricted to paths captured by the specified clock.

If you use the **-rise_from**, **-fall_from**, **-rise_to**, or **-fall_to** option, the report is restricted to only the paths launched or captured by the specified edge type.

EXAMPLES

The following example shows the report when the design has two clocks: *ck1* with a period of 10 and *ck2* with a period of 12. The report shows that the common period is 60 and the multipliers used to obtain the common period are 6 and 5, respectively.

prompt> **report_interclock_relation**

Report : Interlock Relation

A common period of '-' means a full cycle was not expanded.

From	Period1	Count1	To	Period2	Count2	Common
ck1	10.00	6	ck2	12.00	5	60.00
ck2	12.00	5	ck1	10.00	6	60.00

The following example shows the same command using the **-edge** option. The "setup edge" column shows the launch and capture times and the respective edge types (r for rising or f for falling) that result in the smallest setup time requirement. The "hold edge" column shows the same information for the path having the smallest hold time requirement.

prompt> **report_interclock_relation -edge**

Report : Interlock Relation

From	Period1	To	Period2	setup edge	hold edge
				launch capture launch capture	
ck1	10.00	ck2	12.00	10(r) 12(r)	60(r) 60(r)
ck1	10.00	ck1	10.00	0(r) 10(r)	0(r) 0(r)
ck2	12.00	ck1	10.00	48(r) 50(r)	0(r) 0(r)
ck2	12.00	ck2	12.00	0(r) 12(r)	0(r) 0(r)

SEE ALSO

report_clock(2)

report_internal_loads

Displays internal loads on the nets in the current design.

SYNTAX

```
status report_internal_loads  
[-nosplit]
```

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

This command displays all internal loads specified on nets with the **set_load** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following is an example of the internal-loads report:

```
prompt> report_internal_loads  
  
*****  
Report : internal_loads  
Design : comb_internal  
Version: v2.0  
Date   : Fri Nov 30 12:44:23 1990  
*****
```

Attributes:
p - includes pin load

Net	Load	Attributes

n1	3.0000	p
n2	4.5000	

SEE ALSO

report_design(2)
set_load(2)

report_isolate_ports

Displays the status of port isolation on ports on which isolation was requested.

SYNTAX

```
status report_isolate_ports  
[-nosplit]
```

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line in the same column.

DESCRIPTION

The **report_isolate_ports** command displays all ports on which isolation was requested and on which isolation cells were inserted.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of the report that is generated when you run the **report_isolate_ports** command:

```
prompt> report_isolate_ports
```

```
report_isolate_ports
```

```
*****
```

```
Report : isolate_ports
```

```
Design : pi1
```

```
Version: 2001.08
```

```
Date   : Mon Apr 23 15:05:21 2001
```

=====				
Port Name	Cell Name	Inst.Name	Type	Forced Insertion
=====				
out	-	-	buffer	no
qout	IVDA	U3	buffer	no
=====				
1				

SEE ALSO

- remove_isolate_ports(2)
- report_compile_options(2)
- set_isolate_ports(2)

report_isolation_cell

Displays information about isolation cells in the current scope.

SYNTAX

```
status report_isolation_cell
  [isolation_cells]
  [-domain power_domains]
  [-isolation_strategy isolation_strategy_names]
  [-ports pins_ports]
  [-verbose]
  [-nosplit]
```

Data Types

<i>isolation_cells</i>	list
<i>power_domains</i>	list
<i>isolation_strategy_names</i>	string
<i>pins_ports</i>	list

ARGUMENTS

isolation_cells

Specifies the isolation cells that are to be reported. If the specified supply cells do not exist in the current scope, the command fails.

By default, all isolation cells in the current scope are reported.

-domain *power_domains*

Specifies the power domains on which to report all isolation cells. If the power domains specified do not exist in the current scope, the command fails.

-isolation_strategy *isolation_strategy_names*

Specifies the names of the isolation strategies on which to report all isolation cells having the specified isolation strategy. When specifying the isolation strategy, a single power domain must be specified. If the specified isolation strategy is not a part of the specified power domain, the command fails.

-ports *pins_ports*

Reports isolation cells that are isolating the specified pins and ports.

-verbose

Reports the isolation strategy information in addition to the isolation cells information.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The *report_isolation_cell* command enables you to get detailed information about all isolation cells within the current scope. If the **-domain** option is specified, all isolation cells in the specified domains are reported. If the **-isolation_strategy** option is specified, all isolation cells under the specified strategy for the given power domain are reported.

By default, the report displays the following information:

- The name of the port and pin isolated by the isolation cell
- The direction of the port
- The instance name of the isolation cell
- The library reference cell name of the isolation cell.

If you specify the **-verbose** option, the report also displays details of the isolation strategy:

- The name of the isolation strategy
- The location of the cell dictated by the strategy
- The enable signal
- The sense value
- The clamp value
- The elements to which the strategy applies
- The No Isolation information
- The power and ground nets of the strategy.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports on all of the isolation cells in the current scope:

```
prompt> report_isolation_cell
```

```
-----
```

Power Domain : bot

```
=====
| Port | Port Dir.| ISO Cell Name | ISO Lib Cell   | ISO Strategy   |
=====
| DI  | In    | DI_UPF_ISO  | GTECH_ISO0_EN1 | iso           |
=====
```

```
-----
1
```

The following example reports isolation strategy details in addition to the isolation cell information:

prompt> **report_isolation_cell -verbose**

```
-----

Power Domain : bot

=====
| Port | Port Dir.| ISO Cell Name | ISO Lib Cell   | ISO Strategy   |
=====
| DI  | In    | DI_UPF_ISO  | GTECH_ISO0_EN1 | iso           |
=====

=====
|ISO  | Location| Enable| Enable| Clamp| Applies | No    | ISO | ISO |
|Strategy|      | Signal| Sense | Value| to/   | Isolation| Power| Ground |
|      |      |      |      |      |      | net | net |
=====
|iso  | parent | EN   | low  | 0    | -    | -    | iso | -    |
=====
1
```

SEE ALSO

report_level_shifter(2)
 report_retention_cell(2)
 set_isolation(2)
 set_isolation_control(2)

report_keepout_margin

Reports keepout margins of a specified type for the specified cells in the design.

SYNTAX

```
int report_keepout_margin  
  [-type hard | soft]  
  [-original]  
  [-parameters]  
  [-all_derivable]  
  [object_list]
```

Data Types

object_list list

ARGUMENTS

-type hard | soft

Specifies the type of keepout to be reported. The valid values are **hard** or **soft**. By default, keepouts of both soft and hard types are reported for the cells or library cells.

-original

Reports the original keepout margin values which are set using `set_keepout_margin` command and orientation of the cell when the keepout margin is set on the cell.

-parameters

Reports the following pin count based parameter values.

1. `tracks_per_macro_pin`
2. `min_padding_per_macro`
3. `max_padding_per_macro`

-all_derivable

Reports all the derivable keepout margin values for the cells specified in the object list. Keepout margin values set on the cells, derivable from the lib cells and pin count based values in case of macros.

object_list

Describes the list of cells or library cells for which keepouts are to be reported. Keepouts are reported for all cells with `FIXED_PLACEMENT` or library cells in specified in *object_list*.

DESCRIPTION

This command reports keepouts for the specified cells or library cells. If no hard or soft keepout exists the specified cell or library cell, and this command is called to report on it, a warning message is printed out and no action is taken. This command prints out the margin parameters for a keepout. These are of the form {left_x, bottom_y, right_x, top_y}. All units are in microns.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports hard keepouts for cells in an object list named *MY_CELL*.

```
prompt> report_keepout_margin -type hard MY_CELL
```

The following example reports hard keepouts for library cells in an object list named *MY_LIB_CELL*.

```
prompt> report_keepout_margin -type hard MY_LIB_CELL
```

The following example reports the original hard keepout margin values set on the cell in object list named *MY_CELL*.

```
prompt> report_keepout_margin -type hard MY_CELL -original
```

The following example reports all the pin count based parameters which will be used during the calculation of pin count based keepout margins for macro cells

```
prompt> report_keepout_margin -parameters
```

The following example reports all the derivable keepout margin values for the cells in object_list named *MY_CELL*

```
prompt> report_keepout_margin MY_CELL -all_derivable
```

SEE ALSO

`remove_keepout_margin(2)`

report_latch_loop_groups

Reports the latch data pins involved in loops of transparent latches.

SYNTAX

```
list report_latch_loop_groups  
  [-of_objects pin_list]  
  [-loop_breakers_only]  
  [-path_breakers_only]  
  [-nosplit]
```

Data Types

pin_list list

ARGUMENTS

-of_objects *pin_list*

A collection of data pins of transparent latches. Only pins of loop groups containing the specified pins are reported. By default, the command reports all latch data pins involved with transparent latch loops, as well as latch data pins outside of loops that have their **is_latch_loop_breaker** attribute set to **true**.

-loop_breakers_only

Restrict reporting to only the pins that are loop-breaker latch data pins actually contained in transparent latch loops.

-path_breakers_only

Restricts reporting to only the pins that have their **is_latch_loop_breaker** pin attribute set to **true**. By default, all transparent latch data pins within the loop group are reported.

-nosplit

Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The **-nosplit** option prevents line-splitting and facilitates writing software to extract information from the report output.

DESCRIPTION

When sequential loops of transparent latches exist in a design, a loop group containing all the latch data pins of intersecting loops is formed. The command **report_latch_loop_groups** analyzes the design and reports the latch data pins involved with loops. The

report also includes some latch data pins outside loops. Combinational pins are not included in the results.

Latch data pins outside of loops can be reported if the latch is being treated by the tool as a latch loop-breaker data pin, even if it is not inside a latch loop. This can happen when you request that the pin be treated this way using the **set_latch_loop_breakers** command, or if the tool has selected the pin to be treated as a loop-breaker data pin to save runtime. For more information about how this can happen, see the man page for the **timing_through_path_max_segments** variable.

The report lists the name of the latch data pin, a number identifying the latch loop group to which the pin belongs, and attributes describing whether the pin is a loop breaker latch, and why.

The number identifying the latch loop group is an arbitrary number. Within a given report, the latch data pins that are part of the same loop group report the same group number. For latch data pins that are not part of a loop group, "NA" is shown instead of a loop group number.

This command has no effect if the variable **timing_enable_through_paths** is not enabled.

EXAMPLES

The following example uses **report_latch_loop_groups** to report all latch loops in the design:

```
prompt> report_latch_loop_groups
*****
```

```
Report : latch_loop_groups
```

```
Design : test
```

```
Version: 2012.06
```

```
Date   : Mon Aug  4 09:51:19 2012
```

```
*****
```

```
Attributes
```

```
  b loop breaker d pin
```

```
  p long path breaker d pin, but not in a loop
```

```
  u user requested to be a loop breaker using set_latch_loop_breaker
```

```
  a user requested to avoid with set_latch_loop_breaker -avoid
```

Latch D pin	Latch Loop Group	Attributes

DUT/Latch1/D	NA	pu
DUT/Latch2/D	1	b
DUT/Latch3/D	1	
DUT/Latch4/D	2	bu
DUT/Latch5/D	2	a

SEE ALSO

set_latch_loop_breakers(2)

get_latch_loop_groups(2)

timing_enable_through_paths(3)

timing_through_path_max_segments(3)

report_level_shifter

Displays information about level-shifter cells in the current scope.

SYNTAX

```
status report_level_shifter
    [level_shifter_cells]
    [-domain power_domains]
    [-verbose]
    [-nosplit]
    [-macro]
```

Data Types

```
level_shifter_cells  list
power_domains       list
```

ARGUMENTS

level_shifter_cells

Specifies the level-shifter cells that are to be reported. If the specified supply cells do not exist in the current scope, the command fails. By default, all level-shifter cells in the current scope are reported.

-domain *power_domains*

Specifies the power domains to report all level-shifter cells in the design belonging to the power domains. If the power domain specified does not exist in the current scope, the command fails. This argument is optional and can be used to selectively report level-shifter cells.

-verbose

Reports the level-shifter strategies along with the level-shifter cells.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_level_shifter** command enables you to get detailed information about all level-shifter cells in the current scope. If the **-domain** argument is specified, all level-shifter cells in the specified domains are reported. The report consists of two or three parts, depending on whether or not the **-verbose** option is specified.

- The first part contains the level-shifter summary check for a power domain. The summary reports the number of violating level-shifter cells that are marked **dont_touch** and those that are not marked **dont_touch**. It also reports the total number of level-shifter cells belonging to the power domain.
- If the **-verbose** option is specified, the second part of the report provides the details of the level-shifter strategy. The details include the name, type, applies_to/elements, threshold voltage, location, and no shift of the strategy.
- The third part of the report shows the following information about the level-shifter cells:
 - Name of the level-shifter cell
 - Reference name
 - Input and output power net and ground net information, including name and voltage
 - Main power and ground net name and direction
 - Whether or not there is a violation and the reason for the violation
 - Input and output ranges of the level shifter.

Only one range is reported if the input and output ranges are the same.

 - Type of level shifter.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example reports the information about all level-shifter cells in the current scope:

```
prompt> report_level_shifter
```

```
-----
Power Domain : PDT
```

```
*****
```

```
Level shifter check summary
```

```
*****
```

```
No. of violating level shifters - dont_touch : 0
```

```
No. of violating level shifters - no dont_touch : 0
```

```
No. of level shifters in domain : 4
```

```
*****
```

```
=====
Level Shifter   Reference   Input P/G   Output P/G   Main P/G   Violation Reason Voltage Type
                Net Info   Net Info   Net Info
                (Volt)   (Volt)   (Dir.)
=====
```

```
sum_outi[3]_UPF_LS lvi_shifter_2 PNC(0.86) PNT(1.08) PNT(Output) FALSE - --- LH
```



```

          PGT(0.00) PGT(0.00) VSS(Output)
sum_outi[2]_UPF_LS lvi_shifter_2 PNC(0.86) PNT(1.08) PNT(Output) FALSE - --- LH
          PGT(0.00) PGT(0.00) VSS(Output)
sum_outi[1]_UPF_LS lvi_shifter_2 PNC(0.86) PNT(1.08) PNT(Output) FALSE - --- LH
          PGT(0.00) PGT(0.00) VSS(Output)
sum_outi[0]_UPF_LS lvi_shifter_2 PNC(0.86) PNT(1.08) PNT(Output) FALSE - --- LH
          PGT(0.00) PGT(0.00) VSS(Output)

```

=====

1

The following example reports information about all level shifter cells in the current scope along with information about the level shifter strategies:

prompt> **report_level_shifter -verbose**

Power Domain : PD_BOT

Level shifter check summary

No. of violating level shifters - dont_touch : 0

No. of violating level shifters - no dont_touch : 3

No. of level shifters in domain : 3

=====

Level shifter	Strategy	Applies To/ Strategy name	Threshold Elements	Location	No shift Voltage

=====

levshi_bot_1	Always	Both	NA	Inside	FALSE

=====

=====

Level Shifter	Reference Net	Input P/G Info	Output P/G Info	Main P/G	Violation Reason	Voltage	Type
		Net Info (Volt)	Net Info (Volt)	Net Info (Dir.)			

=====

A_UPF_LS	LVLH8X8	MID_VDD(0.90) VSS(0.00)	BOT_VDD(1.08) VSS(0.00)	MID_VDD(Input) VSS(Output)	power mismatch	TRUE	main 0.50-1.50 LH
B_UPF_LS	LVLH8X8	MID_VDD(0.90) VSS(0.00)	BOT_VDD(1.08) VSS(0.00)	MID_VDD(Input) VSS(Input)	power mismatch	TRUE	main 0.50-1.50 LH
Y_UPF_LS	LVLH8X8	BOT_VDD(1.08) VSS(0.00)	MID_VDD(0.90) VSS(0.00)	BOT_VDD(Output) VSS(Both)	power mismatch	TRUE	main 0.50-1.50 HL

=====

1

SEE ALSO

```
report_isolation_cell(2)
report_retention_cell(2)
set_level_shifter(2)
```

report_lib

Displays information about the specified logic library, physical library, or symbol library.

SYNTAX

```
status report_lib
  [-all]
  [-ccs_recv]
  [-em]
  [-fpga]
  [-k_factors]
  [-power]
  [-power_label]
  [-table]
  [-full_table]
  [-timing]
  [-timing_arcs]
  [-timing_label]
  [-noise]
  [-vhdl_name]
  [-yield]
  [-switch]
  [-pg_pin]
  [-char]
  [-operating_condition]
  [-op_cond_name op_cond_name]
  [-routing_rule]
  [-rwm]
  [-user_defined_data]
  library_name
  [cell_list]
  [-jcr]
  [-pattern_must_join_pin]
  [-pattern_must_join_pin_exclusion_list lib_cell_pin_list]
  [-noise_arcs]
  [-multibit]
```

Data Types

```
op_cond_name  string
library_name string
cell_list    string
```

ARGUMENTS

-all

Lists all timing-related, power-related, electromigration-related, and FPGA-related information.

This option applies only to logic libraries.

-ccs_recv

Lists CCS and receiver information for library cells.

This option applies only to logic libraries.

-em

Lists all detailed electromigration-related information.

This option applies only to logic libraries.

-fpga

Lists all detailed FPGA-related information, such as parts and I/O cell attributes.

This option applies only to FPGA libraries.

-k_factors

Lists scaling information for library cells.

This option applies only to logic libraries.

-power

Lists all detailed power-related information.

This option applies only to logic libraries.

-power_label

Lists the power labels for all power arcs, if specified, in a tabular form.

This option applies only to logic libraries.

-table

Lists each cell's state-table description in compact form.

This option applies only to logic libraries.

-full_table

Lists each cell's state-table description in tabular, expanded form.

This option applies only to logic libraries.

-timing

Lists all detailed timing-related information.

This option applies only to logic libraries.

-timing_arcs

Lists all cell timing arcs.

This option applies only to logic libraries.

-timing_label

Lists the timing labels for all timing arcs, if specified, in a tabular form.

This option applies only to logic libraries.

-noise

Lists all detailed noise-related information in the specified library.

This option applies only to logic libraries.

-vhdl_name

Lists the cells and ports whose database (.db) names are different from their VHDL names.

This option applies only to logic libraries.

-yield

Lists failure rate information for library cells.

This option applies only to logic libraries.

-switch

Lists pin switch function information and cell-level steady state current information.

Use the **-pg_pin** option to display the switch function of a power or ground pin.

This option applies only to logic libraries.

-pg_pin

Lists power and ground pin information, such as the PG pin definition and the voltage name to which a signal pin has been linked and the PG type.

This option applies only to logic libraries.

-char

Lists cell characterization information, such as sensitization and pre-driver model.

This option applies only to logic libraries.

-operating_condition

Lists all library operating condition information.

When this is the only option selected, the **report_lib** command reports only operating condition information; no other information is reported.

-op_cond_name *op_cond_name*

Lists operating condition information by name.

When this is the only option selected, the **report_lib** command reports only operating condition information, no other information is reported.

-routing_rule

Lists nondefault routing rule information from physical libraries.

This option applies only to physical libraries.

-rwm

Lists routing wire model information from physical libraries.

In a routing wire model report, the original routing wire model is printed out, which includes overlap wire ratio, adjacent wire ratio, wire ratio and wire length of both directions. Also reported is capacitance (in database units) per micron for each layer and for each direction (based on each layer's wire ratio), based on the default operating condition. The report shows resistance per square (in database units) for each direction based on the default operating condition. These derived numbers provide useful information before applying the routing wire-model using the **set_routing_wire_model** command.

This option applies only to physical libraries.

-user_defined_data

Lists detailed information about the user-defined groups and attributes.

This option applies to both logic libraries and physical libraries.

library_name

Specifies the name of the library to report.

This argument is required.

cell_list

Specifies a list of cells about which information is to be reported. The default is to report information about all cells in the logic library or physical library.

This option applies to both logic libraries and physical libraries.

-jcr

Lists the Job Completion Record information stored in the library.

This option applies only to logic libraries.

-pattern_must_join_pin

Report pattern must-join related items.

-pattern_must_join_pin_exclusion_list *lib_cell_pin_list*

Specifies the collection of lib_pins that don't need the pattern must-join report. This option needs to go with -pattern_must_join_pin.

-noise_arcs

Lists all cell noise arcs.

This option applies only to logic libraries.

DESCRIPTION

The **report_lib** command displays information about the specified logic library, physical library, or symbol library.

By default, a logic library report displays a list of operating conditions, timing ranges, and wire load models, as well as a listing of all

the cells in the library annotated with their attributes. You can restrict the set of cells by using the *cell_list* argument.

Any library object added by using the **update_lib** command is marked with an asterisk (*) following its name, to identify those objects that have been added since the initial library was created with the **read_lib** command.

The **-timing**, **-noise**, **-table**, **-full_table**, **-timing_label**, **-power_label**, **-em**, **-power**, **-yield**, **-pg_pin**, and **-switch** options can be used only with a logic library that was read in from a library source (.lib) file. If the library is read in from a database (.db) file, an error message is issued.

A physical library report displays a list of available layers, a list of vias, a list of sites and a list of cells. If a *cell_list* is specified with **report_lib**, the report includes only the specified cells; otherwise all cells in the given physical library are listed.

A symbol library report displays a list of the names of symbol definitions contained in the library *library_name*. It also reports the route grid and meter scale for the library.

To generate a report, the specified library must be loaded into memory, unless it is found in the search path. The **list_libs** command displays the libraries that are currently loaded.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command generates a report for the *mylib* logic library. This type of report includes names of the library cells, wire load models, timing ranges, and operating conditions, the date the library was created, and the date the library was generated.

```
prompt> report_lib mylib
```

```
*****
Report : library
Library: mylib
Version: C-2009.06
*****
```

```
Library Type      : Technology
Tool Created      : 2003.06
Date Created      : 21-Jun-101 (INF CREATED ON 1-JUL-2001)
Library Version   : 1.000000
Time Unit         : 1ns
```

```
Capacitive Load Unit : 1.000000pf
Pulling Resistance Unit : 1kilo-ohm
Voltage Unit         : 1V
Current Unit         : 1uA
Dynamic Energy Unit  : 1.000000pJ (derived from V,C units)
Leakage Power Unit   : 1pW
Bus Naming Style     : %s[%d] (default)
```

Operating Conditions:

```
Operating Condition Name : mylib
Library : mylib
Process : 1.20
```

Temperature : 125.00
 Voltage : 1.08
 Interconnect Model : worst_case_tree

Input Voltages:

No input_voltage groups specified.

Output Voltages:

No output_voltage groups specified.

default_wire_load_capacitance: 0.000096
 default_wire_load_resistance: 0.000380
 default_wire_load_area: 0.010000

Wire Loading Model:

Name : 05x05
 Location : mylib
 Resistance : 0.00038
 Capacitance : 9.6e-05
 Area : 0.01
 Slope : 15.9634
 Fanout Length Points Average Cap Std Deviation

```
-----
1  9.80
2  22.01
3  35.15
4  49.29
5  64.54
6  80.97
7  98.68
8  117.77
9  138.32
10 160.41
11 184.15
12 209.61
13 236.90
14 266.09
15 297.29
16 330.57
17 366.04
18 403.77
19 443.86
20 486.41
```

...

Wire Loading Model Selection Group:

Name : a

Selection	Wire load name
min area	max area

0.00	5.00	05x05
5.00	10.00	10x10
10.00	20.00	20x20
20.00	30.00	30x30

Wire Loading Model Mode: enclosed.

Wire Loading Model Selection Group: a.

Porosity information:

No porosity information specified.

Routing Layers:

"metal1" "metal2" "metal3" "metal4" "metal5"
"metal6" "metal7" "metal8"

default_min_porosity: 0.000000

In_place optimization mode: match_footprint

Timing Ranges:

No timing ranges specified.

Delay Threshold Trip-Points:

input_threshold_pct_rise: 50
output_threshold_pct_rise: 50
input_threshold_pct_fall: 50
output_threshold_pct_fall: 50

Slew Threshold Trip-Points:

slew_lower_threshold_pct_rise: 20
slew_upper_threshold_pct_rise: 80
slew_lower_threshold_pct_fall: 20
slew_upper_threshold_pct_fall: 80

slew_derate_from_library: 0.6

Components:

Attributes:

af - active falling
ah - active high
al - active low
ar - active rising
b - black box (function unknown)
ce - clock enable
cg - clock gating integrated cell
ub - unbuffered
pgt - pass gate
ctl - has CTL test model
d - dont_touch

iso - isolation cell
 ls - level shifter
 mo - map_only
 1p0g - power only, no ground
 0p1g - ground only, no power
 0p0g - no power/ground
 p - preferred
 hp - hold_preferred
 pwrg - power gating cell
 r - removable
 s - statetable
 sa0 - dont_fault stuck-at-0
 sa1 - dont_fault stuck-at-1
 sa01 - dont_fault both stuck-at-0 and stuck-at-1
 sz - use_for_size_only
 t - test cell. t(scan_type) as
 t(mux_ff) - muxed flip-flop
 t(mux_ld) - muxed latch
 t(clk_scan) - clocked scan
 t(clk_scan_ld) - clocked scan latch
 t(lssd_dld) - double latch LSSD
 t(lssd_sld) - single latch LSSD
 t(lssd_clk) - clocked LSSD
 t(lssd_auxclk) - auxiliary clock LSSD
 u - dont_use
 udp - usable for datapath generators
 ufc - user_function_class. ufc(type) as
 ufc(u) - user-defined
 ufc(a) - automatically generated
 imc - macro cell
 switch_cg - switch cell (coarse grain)
 ret - retention cell
 ao - always on cell
 gicg - generic integrated clock gating
 va - variation-aware ccs timing modeling
 val - variation-aware ccs leakage power modeling
 fil - filler cell
 tap - tap cell
 decap - decap cell
 pll - pll cell

Pin Attributes:

ao - always on pin
 lse - level shifter enable pin
 isoe - isolation cell enable pin
 scmr - standard cell main rail

Cell	Footprint	Attributes
AN2		
OR2		
...		
1		

The following command generates a logic library report that lists all CCS and receiver information for the library cells:

```
prompt> report_lib mylib -ccs_recv
```

```
*****
Report : library
Library: mylib
Version: C-2009.06
*****
```

...

Components:

Attributes:

```
af - active falling
ah - active high
al - active low
...
ccs - composite current source
c_ccs - compact ccs timing modeling
ccsn - ccs noise data modeling
recv - receiver model
...
tap - tap cell
decap - decap cell
pll - pll cell
```

Pin Attributes:

```
ao - always on pin
lse - level shifter enable pin
isoe - isolation cell enable pin
scmr - standard cell main rail
```

Cell	Footprint	Attributes
AN2		ccs, recv
OR2		ccs

...

1

The following command generates a report on library timing in the *AND2* cell of the *tech_lib* logic library:

```
prompt> report_lib tech_lib -timing AND2
```

```
*****
Report : library
Library: tech_lib
Version: C-2009.06
*****
```

```
Library Type      : Technology
Tool Created      : 2003.06
Date Created      : 21-Jun-101 (INF CREATED ON 1-JUL-2001)
Library Version   : 1.000000
```

```
Delay Model       : table_lookup
Time Unit         : 1ns
```

Timing Attributes Defaults:

Attribute	Default

```

-----
max_transition
max_fanout
fanout_load      1
max_capacitance

```

Lookup Table Template:

```

Template_name
-----
del_0_5_7_w
  VARIABLE_1: input_net_transition
  VARIABLE_2: total_output_net_capacitance
  INDEX_1: 0.0150 0.2500 0.6500 1.4000 3.0000
  INDEX_2: 0.0000 0.0035 0.0070 0.0192 0.0402 0.0752 0.1750
...

```

CELL(AND2): 1.25;

```

PIN(A): in, 0.0035, 1, 3, , ;
END_PIN A;

```

```

PIN(B): in, 0.00375, 1, 3, , ;
END_PIN B;

```

```

PIN(Z): out, 0, 18, 2, 0.175, , ;
DELAY: A, Z, prop, pos_unate, " , ( , ) , ( , ) , ( , ) ;
cell_rise ( del_0_5_7_w ) :
  VALUES:  0.0710 0.1000 0.1240 0.2050 0.3440 0.5750
            1.2330 0.1120 0.1410 0.1650 0.2470 0.3860
            0.6170 1.2750 0.1520 0.1840 0.2100 0.2920
            0.4320 0.6630 1.3210 0.1990 0.2360 0.2630
            0.3470 0.4870 0.7190 1.3770 0.2630 0.3080
            0.3400 0.4280 0.5700 0.8030 1.4630

```

```

cell_fall ( del_0_5_7_w ) :
  VALUES:  0.0620 0.0900 0.1160 0.2020 0.3500 0.5950
            1.2940 0.1050 0.1360 0.1620 0.2480 0.3960
            0.6420 1.3410 0.1440 0.1770 0.2040 0.2930
            0.4410 0.6860 1.3860 0.1870 0.2240 0.2530
            0.3430 0.4920 0.7390 1.4380 0.2410 0.2850
            0.3180 0.4130 0.5650 0.8140 1.5150

```

```

rise_transition ( del_0_5_7_w ) :
  VALUES:  0.0450 0.0960 0.1480 0.3400 0.6720 1.2270
            2.8080 0.0490 0.0980 0.1500 0.3400 0.6720
            1.2270 2.8080 0.0580 0.1060 0.1560 0.3440
            0.6740 1.2270 2.8080 0.0720 0.1200 0.1660
            0.3480 0.6770 1.2300 2.8080 0.0930 0.1460
            0.1910 0.3630 0.6860 1.2370 2.8120

```

```

fall_transition ( del_0_5_7_w ) :
  VALUES:  0.0370 0.0900 0.1440 0.3380 0.6720 1.2300
            2.8190 0.0430 0.0930 0.1460 0.3380 0.6720
            1.2300 2.8200 0.0510 0.1020 0.1530 0.3420
            0.6740 1.2300 2.8190 0.0650 0.1150 0.1640

```

```

0.3480 0.6780 1.2330 2.8200 0.0870 0.1400
0.1870 0.3660 0.6920 1.2430 2.8250

```

```

DELAY: B, Z, prop, pos_unate, " , ( , ) , ( , ) , ( , ) ;

```

```

...

```

```

END_PIN Z;
END_CELL AND2;

```

```

Number of library cells: 1
Number of reported cells: 1
1

```

The following command generates a report on library noise in the *AND2* cell of the *tech_lib* logic library:

```

prompt> report_lib tech_lib -noise AND2

```

```

...

```

```

CELL(AND2): 4, ;

```

```

PIN(A): in, 2, , , ;
END_PIN A;

```

```

PIN(B): in, 2, , , ;
END_PIN B;

```

```

PIN(Z): out, , , , , ( , ) , ( , ) ;

```

```

noise_immunity_high ( noise5x5 ) :

```

```

INDEX_2 : 0.3620 0.7250 1.0870 1.4490 1.8120

```

```

VALUES : 0.7330 1.0730 1.4120 1.7520 2.0920 0.8730

```

```

1.2140 1.5540 1.8940 2.2340 1.0950 1.4420

```

```

1.7870 2.1320 2.4770 1.2980 1.6480 1.9960

```

```

2.3430 2.6910 1.7030 2.0600 2.4140 2.7660

```

```

3.1190

```

```

END_PIN Z;
END_CELL AND2;

```

For details about library timing report contents and syntax, see the *Library Compiler Modeling Timing, Signal Integrity, and Power in Technology Libraries User Guide*.

The following example generates a report on power information in the *tech_lib* logic library:

```

prompt> report_lib tech_lib -power

```

```

...

```

```

Power Information:

```

```

Attributes:

```

```

a - average power specification

```

```

i - internal power

```

```

l - leakage power

```

```

rf - rise and fall power specification

```

```

Power

```

```

Cell      # Attr  Toggling pin  source of path  When
-----

```

```

AN2       0 l
          1 i,a Z      A

```

```

      2 i,a  Z    B
INV    0 I
      1 i,a  Z    A
NO2    0 I
      1 i,a  Z    A
      2 i,a  Z    B
flop1  0 I
      1 i,a  CP
      2 i,a  Q    CP
      3 i,a  Q    D
      4 i,a  QN   CP
      5 i,a  QN   D
latch1 0 I
      1 i,a  G
      2 i,a  Q    G
      3 i,a  Q    D
      4 i,a  QN   G
      5 i,a  QN   D

```

The following command generates a report on library electromigration in the *ad3* cell of the *tech_lib* logic library:

```

prompt> report_lib tech_lib -em ad3
...
CELL(ad3): 2;
PIN(y): out, 0, , , 1.812, , ;
ELECTROMIGRATION: a;
  em_max_toggle_rate ( output_by_cap_and_trans ) :
    VALUES: 2.0000 1.0000 0.5000 1.5000 0.7500 0.3300
              1.0000 0.5000 0.1500

ELECTROMIGRATION: b;
  em_max_toggle_rate ( output_by_cap_and_trans ) :
    VALUES: 2.0000 1.0000 0.5000 1.5000 0.7500 0.3300
              1.0000 0.5000 0.1500

ELECTROMIGRATION: c;
  em_max_toggle_rate ( output_by_cap_and_trans ) :
    VALUES: 2.0000 1.0000 0.5000 1.5000 0.7500 0.3300
              1.0000 0.5000 0.1500

END_PIN y;

PIN(a): in, 1, , , ;
END_PIN a;

PIN(b): in, 1, , , ;
END_PIN b;

PIN(c): in, 1, , , ;
ELECTROMIGRATION:
  em_max_toggle_rate ( input_by_trans ) :
    VALUES: 1.5000 1.0000 0.5000

END_PIN c;
END_CELL ad3;

```

Number of library cells: 1

Number of reported cells for em: 1

1

For details about library electromigration report contents and syntax, see the *Library Compiler Modeling Timing, Signal Integrity, and Power in Technology Libraries User Guide*.

The following command generates a report on timing arcs in the *tech_lib* logic library. The *INV_NEW* cell is identified as having been added using the **update_lib** command:

```
prompt> report_lib tech_lib -timing_arcs
```

...

Cell	Arc		Arc Pins		From	To	When
	Attributes	#	Sense/type				
AND		0	pos unate	A	Z		
		1	pos unate	B	Z		
INVET		0	neg unate	A	Z		
INV_NEW		0	neg unate	A	Z		

...

If the **-timing_arcs** option is not used, the last five columns do not appear.

The following command generates a report for a symbol library. The report includes the names of all symbol definitions, the *route_grid*, and the *meter_scale*:

```
prompt> report_lib sym_lib.sdb
```

The following command generates a report that displays timing arcs for the inverter cell from the *tech_lib* logic library:

```
prompt> report_lib tech_lib -timing_arcs inverter
```

The following example generates a report that displays noise arcs for the inverter cell from the *tech_lib* logic library:

```
prompt> report_lib tech_lib -noise_arcs inverter
```

The following is an example of a compact report on state-table information in the *tech_lib* logic library:

```
prompt> report_lib tech_lib -table
```

State table descriptions:

CELL(D_LATCH):

PIN(Q): CD, D, G

TABLE: HNLNLLLL

END_PIN Q;

PIN(QN):

STATE_FUNCTION: Q'

END_PIN QN;

END_CELL D_LATCH;

The following is an example of a tabular report on state-table information in the *tech_lib* logic library:

```
prompt> report_lib tech_lib -full_table
```

State table descriptions:

CELL(D_LATCH):

```
PIN(Q): CD, D, G
```

```
TABLE:
```

```
000 L
```

```
001 L
```

```
010 L
```

```
011 L
```

```
100 N
```

```
101 L
```

```
110 N
```

```
111 H
```

```
END_PIN Q;
```

```
PIN(QN):
```

```
STATE_FUNCTION: Q'
```

```
END_PIN QN;
```

```
END_CELL D_LATCH;
```

The following is an example of a pg_pin report in the *tech_lib* logic library:

```
prompt> report_lib tech_lib -pg_pin
```

```
...
```

```
CELL(LVLHLEHX2):
```

```
PG_PIN(VDDI):
```

```
VOLTAGE_NAME: VDDH
```

```
PG_TYPE: primary_power
```

```
END_PIN VDDI;
```

```
PG_PIN(VSS):
```

```
VOLTAGE: VSS
```

```
PG_TYPE: primary_ground
```

```
END_PIN VSS;
```

```
PIN(QN):
```

```
RELATED_POWER_PIN : VDDI
```

```
RELATED_GROUND_PIN : VSS
```

```
END_PIN QN;
```

```
END_CELL LVLHLEHX2;
```

The following is an example of a characterization report in the *tech_lib* logic library:

```
prompt> report_lib tech_lib -char
```

```
*****
```

```
Report : library
```

```
Library: tech_lib
```

```
Version: A-2007.12
```

```
*****
```

```
Library Type      : Technology
```

```
Tool Created      : A-2007.12
```

```
Date Created      : Not Specified
```

```
Library Version    : Not Specified
```



```

Sensitization : 2IN_1OUT
pin_names : (A, B, C)
vector(0) : (0, 0, 0)
vector(1) : (0, 0, 1)
vector(2) : (0, 1, 0)
vector(3) : (0, 1, 1)
vector(4) : (1, 0, 0)
vector(5) : (1, 0, 1)
vector(6) : (1, 1, 0)
vector(7) : (1, 1, 1)
vector(8) : (1, 1, 1)

CELL(AN2): 2;
SENSITIZATION_MASTER: 2IN_1OUT
PIN_NAME_MAP : (A, B, Z)

PIN(A): in, 1, , , , ;
END_PIN A;

PIN(B): in, 1, , , , ;
END_PIN B;

PIN(Z): out, 0, , , , , ;
DELAY: A, Z, prop, pos_unate, ", (1, 1), (0, 0), (0.1443, 0.0523);
sensitization_master: 2IN_1OUT
pin_name_map : (A, B, Z)
wave_rise : (3, 4, 5)
wave_fall : (0, 3, 7)
wave_rise_sampling_index: 2
wave_fall_sampling_index: 2
wave_rise_timing_interval : (0, 0.5)
wave_fall_timing_interval : (0, 0.45)
DELAY: B, Z, prop, pos_unate, ", (0.48, 0.77), (0, 0), (0.1443, 0.0523);
END_PIN Z;
END_CELL AN2;

```

The following is an example of a switch-cell report in the *tech_lib* logic library:

```

prompt> report_lib tech_lib -switch
...
CELL(FD1): 2;

DC_CURRENT:
dc_current ( ccsn_dc_29x29 ) :
RELATED_SWITCH_PIN : D
RELATED_PG_PIN : PWR
RELATED_INTERNAL_PG_PIN : VVDD
INDEX_1 : -1.2000 -0.6000 -0.2400 -0.1200 0.0000 0.0600
          0.1200 0.1800 0.2400 0.3000 0.3600 0.4200
          0.4800 0.5400 0.6000 0.6600 0.7200 0.7800
          0.8400 0.9000 0.9600 1.0200 1.0800 1.1400
          1.2000 1.3200 1.4400 1.8000 2.4000

...
END_CELL AN2;

```

The following is an example of a jcr report in the *tech_lib* logic library:

prompt> **report_lib tech_lib -jcr**

Report : library
Library: z0
Version: I-2013.12-SP
Date : Sun Nov 17 23:08:39 2013

Library Type : Technology
Tool Created : I-2013.12-SP
Date Created : Not Specified
Library Version : Not Specified

JCR Infomation :

#SNPS_JCR_INFORMATION_BEGIN
#VERSION=1.1 JCR_CHKSUM=47660
#PRODUCT_NAME=MyProduct
#PRODUCT_VERSION=2013.06
#PRODUCT_EXEC=my_exe
#HOST=rcrh1
#DESIGN=file1.db
#NUMBER_OF_ERRORS=0
#NUMBER_OF_WARNINGS=0
#NUMBER_OF_INFO_MSG=0
#RUN_STATUS=0
#START_TIME=Mon Nov 04 22:00:21 2013
#END_TIME=Mon Nov 04 22:00:21 2013
#OUTPUT_FOLDER=
#NUMBER_OF_FILES=0
#LOG_FILE_PATH=
#WARNING_AS_ERROR=0
#ERROR_AS_WARNING=0
#NUMBER_OF_THREADS=0
#NUMBER_OF_CORES=0
#MEMORY_UTILIZATION=0
#DOWNSTRAEM_PRODUCT_INFO=(name=star_rcxt), (error=10), (warning=20), (info_msg=0)
#DOWNSTRAEM_PRODUCT_INFO=(name=hspice), (error=30), (warning=50), (info_msg=10)
#NOTES=
#SNPS_JCR_INFORMATION_END

#SNPS_JCR_INFORMATION_BEGIN
#VERSION=1.1 JCR_CHKSUM=41970
#PRODUCT_NAME=dc_shell
#PRODUCT_VERSION=I-2013.12-SP
#PRODUCT_EXEC=
#HOST=madyak669
#DESIGN=z0.lib
#NUMBER_OF_ERRORS=0
#NUMBER_OF_WARNINGS=9
#NUMBER_OF_INFO_MSG=0
#RUN_STATUS=0
#START_TIME=Sun Nov 17 23:08:36 2013
#END_TIME=Sun Nov 17 23:08:39 2013
#OUTPUT_FOLDER=

```
#NUMBER_OF_FILES=0
#LOG_FILE_PATH=
#WARNING_AS_ERROR=0
#ERROR_AS_WARNING=0
#NUMBER_OF_THREADS=0
#NUMBER_OF_CORES=0
#MEMORY_UTILIZATION=0
#NOTES=
#SNPS_JCR_INFORMATION_END
```

SEE ALSO

compare_lib(2)
read_lib(2)
update_lib(2)
write_lib(2)

report_libcell_subset

Reports the target library family specified on sequential and instantiated combinational cells.

SYNTAX

```
status report_libcell_subset  
  [-object_list cells]  
  [-nosplit]
```

Data Types

cells list

ARGUMENTS

-object_list *cells*

Specifies the cells for which to report the target library family.

-nosplit

Specifies not to split lines when column fields overflow.

DESCRIPTION

This command reports information about the target library subset that is specified on the cells.

Note that the command only reports the library subset information that is explicitly user-defined.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the target library family for the u1 sequential cell:

```
prompt> report_libcell_subset -object_list [get_cells u1]
```

Report : library cell subset
Design : chip
Version: G-2012.06
Date : Wed Mar 21 16:16:58 2012

Libcell Family		Libcells

special_flops	sff0 sff1 sff2	

Object	Reference	Libcell Family

u1	sff0	special_flops

1		

SEE ALSO

- remove_libcell_subset(2)
- report_libcell_subset(2)
- set_libcell_subset(2)
- define_libcell_subset(2)

report_link_library_subset

Reports link library subsets on the design.

SYNTAX

```
status report_link_library_subset  
  [-object_list cells]  
  [-top]  
  [-scenarios scenarios]  
  [-nosplit]
```

Data Types

```
cells      collection  
scenarios collection
```

ARGUMENTS

-object_list *cells*

Specifies the cells on which to report the link library subset.

-top

Requests a report of the link library subset on the top-level design.

-scenarios *scenarios*

Specifies the scenarios for which to report the link library subset. Note that the command does not report inactive scenarios.

By default, only the current scenario is reported.

-nosplit

Specifies not to split lines when column fields overflow.

DESCRIPTION

This command reports the link library subset on the design based on the specified options.

If neither the **-object_list** nor **-top** option is specified, all the link library subsets set in the design are reported.

If a cell is specified in the **-object_list** option does not have a subset set directly on it, but there is a subset on a parent instance or

on the top-level design, the report lists the subset in effect on the cell and indicates that it was "Inherited".

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following example reports the link library subset for the top-level design and for the u1 instance:

```
prompt> report_link_library_subset -top -object_list [get_cells u1]
```

SEE ALSO

remove_link_library_subset(2)
set_link_library_subset(2)
link_library(3)

report_logic_levels

Displays logic levels information about a design

SYNTAX

```
status report_logic_levels
  [-to to_list]
  [-from from_list]
  [-through through_list]
  [-exclude exclude_list]
  [-nworst paths_per_endpoint]
  [-max_paths max_path_count]
  [-start_end_pair]
  [-group group_name]
  [-slack_greater_than greater_slack_limit]
  [-slack_lesser_than lesser_slack_limit]
  [-path_type path_type]
  [-sort_paths_by sort_path_value]
  [-num_bins number_of_bins]
  [-bin_width bin_width_number]
  [-max_paths_to_report max_path_report_count]
  [-export_csv_file]
  [-summary_only]
  [-nosplit]
```

Data Types

<i>to_list</i>	list
<i>from_list</i>	list
<i>through_list</i>	list
<i>exclude_list</i>	list
<i>paths_per_endpoint</i>	integer
<i>max_path_count</i>	integer
<i>group_name</i>	string
<i>greater_slack_limit</i>	float
<i>lesser_slack_limit</i>	float
<i>path_type</i>	string
<i>sort_path_value</i>	string
<i>number_of_bins</i>	integer
<i>bin_width_number</i>	integer
<i>max_path_report_count</i>	integer
<i>output_csv_file_name</i>	string

ARGUMENTS

-to *to_list*

Specifies the to pins, ports, nets, or clocks. Path endpoints are the output ports or data pins of registers. If you specify a clock, the command considers all endpoints that are constrained by the clock.

-from *from_list*

Specifies the from pins, ports, nets, or clocks. Path startpoints are the input ports or clock pins of registers. If you specify a clock, the command considers all startpoints clocked by the clock.

-through *through_list*

Reports only paths that pass through the named pins, ports, or clocks.

-exclude *exclude_list*

Excludes all data paths from/through/to the named pins, ports, nets, and cell instances. Any data path that starts from, contains, or ends on a listed object is excluded. If you specify a cell instance, data paths that include any pin of that cell instance are excluded. This option has higher precedence than the **-from**, **-through**, **-to**, and similar options.

This option is not applied to clock pins.

-nworst *paths_per_endpoint*

Specifies the maximum number of paths to get per endpoint. The default is 1, which gets only the single worst path ending at a given endpoint.

-max_paths *max_path_count*

Specifies the maximum number of paths to analyze per path group. The default is 100000.

-start_end_pair

Limits the analysis of paths returned to the single worst timing path per each combination of startpoint and endpoint found. No other paths are included that have the same startpoint and same endpoint.

-group *group_name*

Restricts the analysis to paths in this *group_name*. Use the **group_path** or **create_clock** command to group paths.

-slack_greater_than *greater_slack_limit*

Selects only those paths with a slack greater than *greater_slack_limit*. The **-slack_greater_than** option can be combined with the **-slack_lesser_than** option to select only those paths inside or outside of a given slack range. The specified value is in main library units.

-slack_lesser_than *lesser_slack_limit*

Selects only those paths with a slack less than *lesser_slack_limit*. The **-slack_greater_than** option can be combined with the **-slack_lesser_than** option to select only those paths inside or outside of a given slack range. The specified value is in main library units. The default value for this option is 0.0. By default, the command will analyze only the -ve slack paths due to this default setting.

-path_type *path_type*

Specifies the type of path to be considered for logic level analysis. By default, the all paths are considered.

-sort_paths_by *sort_path_value*

Specifies the criterion on which to sort the paths which will be displayed in the logic level path report section. Only the top paths specified by option **-max_paths_to_report** will be reported after sorting.

-num_bins *number_of_bins*

Specifies the number of bins to be used in the Logic level distribution section. If this option is not specified, the tool uses default value of 10.

-bin_width *bin_width_number*

Specifies the width of the bin to be used in the Logic level distribution section. If both -num_bins & -bin_width options are specified, then only -num_bins is used. When -bin_width option is specified, the tool computes the number of bins based on this width value.

-max_paths_to_report *max_path_report_count*

Specifies the number of paths to be reported in the logic level path report section.

-export_csv_file *output_csv_file_name*

Specifies the file name for the csv file. The entire contents of the report will be exported to the specified csv file in csv (comma separated values) format.

-summary_only

If this option is specified, then only the summary section is printed. Logic level distribution section & Logic level path report for each of the path groups are not reported.

-nosplit

Prevents line splitting and facilitates writing application to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

This command reports the logic levels information about the design.

By default, the command analyses only -ve slack paths from all path groups with a maximum of 100000 paths for each path group.

The command collects the paths according to the user specifications and calculates the logic levels in those paths. Logic levels are the number of mapped cells in the path excluding buffers and invertors. It then compares it against the threshold specified by the user before or automatically computed one and determines whether it violates the threshold or not. It also collects other information about the path such as WNS, start and end point, start and end clocks and reports them.

The report has three sections.

Summary section

In this section, logic level and timing information are reported for cumulative and each of the path groups. The cumulative information is reported as "All path groups".

The columns in this section are customizable through setting of **logic_level_report_summary_format** variable.

GROUP	REQUIRED PERIOD	NUM OF WNS	MAX PATHS	LEVEL
-----	-----	-----	-----	-----
All Path Groups	n/a	n/a	2310	13
CLOCK	0.0100	-2.3289	1600	13

custom_group1	0.0100	-2.3174	166	12
custom_group2	0.0100	-2.3006	544	13

Summary section is followed by Logic level distribution section & Logic level path report section for cumulative and each of the path groups. The cumulative information is reported under the path group name "All path groups".

Logic level distribution section

In this section, logic level distribution of paths for the particular path group are reported. The columns in this section are not customizable and are fixed.

LOGIC LEVELS	NUMBER OF PATHS	PERCENTAGE OF PATHS
0 to 1	4	0%
2 to 3	22	1%
4 to 5	1017	44%
6 to 7	908	39%
8 to 9	47	2%
10 to 11	205	9%
12 to 13	107	5%

Logic level path report

In this section, logic level & timing information for the top paths for the particular path group are reported. The columns in this section are customizable through setting of **logic_level_report_group_format** variable.

WNS	LOGIC LEVELS	THRESHOLD	START POINT	END POINT
-1.3106	13	4	REG_BLK/DATA_OUT_reg[0]/CK	UPC_BLK/DATA_OUT_reg[7]/D
-1.3072	11	4	REG_BLK/DATA_OUT_reg[0]/CK	UPC_BLK/DATA_OUT_reg[3]/D
-1.2999	10	4	REG_BLK/DATA_OUT_reg[0]/CK	UPC_BLK/DATA_OUT_reg[3]/D
-1.2991	11	4	REG_BLK/DATA_OUT_reg[0]/CK	UPC_BLK/DATA_OUT_reg[9]/D
-1.2979	12	4	REG_BLK/DATA_OUT_reg[0]/CK	UPC_BLK/DATA_OUT_reg[5]/D

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example sets the global logic level threshold to 5 and then runs logic level report for path group "CLOCK"

```
prompt> set_analyze_rtl_logic_level_threshold 5
prompt> report_logic_levels -group CLOCK
```

The following example invokes the reporting command with csv file option.

```
prompt> report_logic_levels -export_csv_file ll.csv
```

SEE ALSO

set_analyze_rtl_logic_level_threshold(2)
get_timing_paths(2)
logic_level_report_summary_format(3)
logic_level_report_group_format(3)

report_logicbist_configuration

Displays options specified by the **set_logicbist_configuration** command.

SYNTAX

```
status report_logicbist_configuration  
[-test_mode mode_name_list | all]
```

Data Types

mode_name_list list

ARGUMENTS

-test_mode *mode_name_list* | all

Specifies the test mode(s) to report. If not specified, the tool reports on the *current_test_mode* that can be displayed by the **current_test_mode** command. If no test modes were created, the tool defaults to the default LogicBIST compression mode. If **all** is specified, the tool displays the scan configurations of all LogicBIST compression test modes, including the default test mode (all_dft).

DESCRIPTION

This command displays options specified by the **set_logicbist_configuration** command on the current design.

EXAMPLES

The following is an example of a LogicBIST compression configuration report:

```
prompt> report_logicbist_configuration
```

```
*****  
Report : LogicBIST Configuration  
Design : top  
Version: L-2016.03-VAL  
Date   : Thu Apr 7 05:34:04 2016  
*****
```

```
=====
TEST MODE: my_bist_mode
VIEW   : Specification
=====
Shift Clock           CLK1
Chain Count           96
Maximum Chain Length  Unspecified
PRPG Width            32
MISR Width            32
Shift Counter Width   Unspecified
Pattern Counter Width 12
Burn-in               Disable
Power ramp-up         Disable
```

1

SEE ALSO

- current_test_mode(2)
- reset_logicbist_configuration(2)
- set_logicbist_configuration(2)

report_min_pulse_width

Displays minimum pulse width check information about specified sequential device clock pins.

SYNTAX

```
status report_min_pulse_width
  [-all_violators]
  [-significant_digits digits]
  [-nosplit]
  [-scenarios scenario_list]
  [pin_list]
```

Data Types

```
digits      int
scenario_list list
pin_list    list
```

ARGUMENTS

-all_violators

Indicates that only violating minimum pulse width checks are reported.

-significant_digits *digits*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, which default value is 2. Use this option if you want to override the default.

-nosplit

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the width of the column, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line splitting and facilitates writing software to extract information from the report output.

-scenarios *scenario_list*

Reports min pulse width constraint in scenarios given in the list. If this option is not given, min pulse width constraints on all scenarios are reported.

pin_list

Specifies a list of clock pins of sequential devices to report. By default, the report contains all clock pins in the current design. If non-clock pins are specified, they will be ignored.

DESCRIPTION

The **report_min_pulse_width** command displays the following information for the minimum pulse width check: required pulse width, actual pulse width, and by how much the constraint is violated or met (slack). This information is listed in order of decreasing slack starting with the worst violator.

Minimum pulse width violations can also be reported with the **report_constraint** command.

EXAMPLES

The following example generates a report of all minimum pulse width violations in the current design.

```
prompt> report_min_pulse_width
```

```
***** Report : min_pulse_width Design : test Version: F-2011.09-SP Date : Tue Oct 18 13:24:39
2011 ***** Required Actual Pin pulse width pulse width Slack Scenario -----
----- ff1/cp(low) 2.00 0.87 -1.13(VIOLATED) ff2/cp(low) 2.00 1.41 -0.59(VIOLATED) ff2/cp(high) 0.60 0.59
-0.01(VIOLATED) ff1/cp(high) 0.60 1.13 0.53(MET)
```

SEE ALSO

```
remove_min_pulse_width(2)
report_constraint(2)
set_min_pulse_width(2)
report_default_significant_digits(3)
```

report_missing_constraints

Reports missing constraints in current design.

SYNTAX

```
status report_missing_constraints  
[-class class_list]  
[-clock_tracing_mode tracing_mode]
```

Data Types

```
class_list    list  
tracing_mode string
```

ARGUMENTS

-class *class_list*

Specifies a list of missing constraint classes to be reported. You can enter the following classes:

- clocks
- input_delay
- output_delay
- pin_load
- driving_lib_cell

-clock_tracing_mode *tracing_mode*

Specifies tracing mode for tracing clocks from missing clock pins. You can specify the following modes:

- simple
- advanced

When the option is set to simple (the default), the tool traces only ports as clocks. When the option is set to advanced, the tool traces ports and register outputs as clocks.

DESCRIPTION

This command reports missing constraints of the specified classes in current design. If no class is specified, the command reports all missing constraints of all the classes in current design.

This command is for DC Explore only.

SEE ALSO

`create_missing_constraints(2)`

report_mode

Prints a report of the instance modes.

SYNTAX

```
string report_mode  
  [-nosplit]  
  [instance_list]
```

Data Types

instance_list list

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. By default, most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

instance_list

Indicates that the mode report is to include only the specified cells. By default, all cells that have modes are included.

DESCRIPTION

This command reports which cells have modes and for each mode, specifies that the mode is currently enabled or disabled. This reports reflects the mode specifications of the **set_mode** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example reports that two cells in the design have modes: the *Uram1* cell and the *Uram2* cell.

prompt> **report_mode**

```
*****
Report : mode
Design : top_design
*****
```

Cell	Mode(Group)	Status

Uram1/core(RAM2_core)	read(rw)	ENABLED
	write(rw)	ENABLED

Uram2/core(RAM2_core)	read(rw)	ENABLED
	write(rw)	ENABLED

The following example reports that modes are specified for the 2 RAMs that have modes in the design. Ram *Uram1* is set in mode read, and Ram *Uram2* is set in mode write. This means all timing arcs of RAM *Uram1* associated with mode read are enabled, and all timing arcs associated with mode write are disabled.

prompt> **set_mode read Uram1/core**

prompt> **set_mode write Uram2/core**

prompt> **report_mode**

```
*****
Report : mode
Design : top_design
*****
```

Cell	Mode(Group)	Status

Uram1/core(RAM2_core)	read(rw)	ENABLED
	write(rw)	disabled

Uram2/core(RAM2_core)	read(rw)	disabled
	write(rw)	ENABLED

SEE ALSO

reset_mode(2)
set_mode(2)

report_multibit

Displays information about multibit components in the current design or the subdesign.

SYNTAX

```
status report_multibit  
  [-nosplit]  
  [-hierarchical]  
  [object_list]
```

Data Types

object_list list

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-hierarchical

Report multibit components in all submodules of the specified object. If this option is not specified, do not report multibit components in submodules.

object_list

Specifies a list of cells and multibit components to show in the report. If this option is not specified, all multibit components in the current design are listed. For specified cells, the report is done on multibit components that contain the specified cells.

DESCRIPTION

The **report_multibit** command displays information and statistics about multibit components in the current design or specified instances. A multibit component exists in a design due to the instantiation of a multibit library cell in the RTL, inference from the HDL source, or usage of the **create_multibit** command. A multibit component is a grouping of register cells targeted for possible implementation using a multibit register (or multiple multibit registers) for the register bits of the group.

The data displayed by this command might not be accurate if the design cannot be linked.

EXAMPLES

The following example shows a report on all of the multibit components in the current design:

prompt> **report_multibit**

```
*****
Report : multibit
Design : subtest
Version: 1998.02
Date   : Fri Aug 29 10:01:19 1997
*****
```

Attributes:

b - black box (unknown)
 h - hierarchical
 n - noncombinational
 r - removable
 u - contains unmapped logic

Multibit Component : y_reg

Cell	Reference	Library	Area	Width	Attributes
y_reg[3]	**SEQGEN**		0.00	1	n, u
y_reg[2]	**SEQGEN**		0.00	1	n, u
y_reg[1]	**SEQGEN**		0.00	1	n, u
y_reg[0]	**SEQGEN**		0.00	1	n, u
Total 4 cells			0.00	4	

Multibit Component : z_reg

Cell	Reference	Library	Area	Width	Attributes
z_reg[3]	**SEQGEN**		0.00	1	n, u
z_reg[2]	**SEQGEN**		0.00	1	n, u
z_reg[1]	**SEQGEN**		0.00	1	n, u
z_reg[0]	**SEQGEN**		0.00	1	n, u
Total 4 cells			0.00	4	

Total 2 Multibit Components

The following example shows a report on one multibit component:

prompt> **report_multibit y_reg**

```
*****
Report : multibit
Design : subtest
Version: 1998.02
Date   : Fri Aug 29 10:01:19 1997
*****
```

Attributes:

b - black box (unknown)
 h - hierarchical
 n - noncombinational
 r - removable
 u - contains unmapped logic

Multibit Component : y_reg

Cell	Reference	Library	Area	Width	Attributes
y_reg[3]	**SEQGEN**		0.00	1	n, u
y_reg[2]	**SEQGEN**		0.00	1	n, u
y_reg[1]	**SEQGEN**		0.00	1	n, u
y_reg[0]	**SEQGEN**		0.00	1	n, u
Total 4 cells			0.00	4	

Total 1 Multibit Components

The following example shows a report generated with a cell as an argument:

prompt> **report_multibit y_reg[0]**

```

*****
Report : multibit
Design : subtest
Version: 1998.02
Date   : Fri Aug 29 10:01:19 1997
*****

```

Attributes:

b - black box (unknown)
 h - hierarchical
 n - noncombinational
 r - removable
 u - contains unmapped logic

Multibit Component : y_reg

Cell	Reference	Library	Area	Width	Attributes
y_reg[3]	**SEQGEN**		0.00	1	n, u
y_reg[2]	**SEQGEN**		0.00	1	n, u
y_reg[1]	**SEQGEN**		0.00	1	n, u
y_reg[0]	**SEQGEN**		0.00	1	n, u
Total 4 cells			0.00	4	

Total 1 Multibit Components

The following example shows **report_multibit** with the *y_reg* multibit component from the *U1/BOT1* design instance:

prompt> **report_multibit U1/BOT1/y_reg**

```

*****
Report : multibit
Design : subtest
Version: 1998.02
Date   : Fri Aug 29 10:01:19 1997

```

Attributes:

b - black box (unknown)
 h - hierarchical
 n - noncombinational
 r - removable
 u - contains unmapped logic

Multibit Component : y_reg

Cell	Reference	Library	Area	Width	Attributes
U1/BOT1/y_reg[3]	**SEQGEN**		0.00	1	n, u
U1/BOT1/y_reg[2]	**SEQGEN**		0.00	1	n, u
U1/BOT1/y_reg[1]	**SEQGEN**		0.00	1	n, u
U1/BOT1/y_reg[0]	**SEQGEN**		0.00	1	n, u
Total 4 cells			0.00	4	

Total 1 Multibit Components

The following example shows hierarchically report multibit components in the instance *mid0*:

prompt> **report_multibit -hierarchical mid0**

Report : multibit

Design : top

Version: G-2012.06-ALPHA4

Date : Fri Mar 2 14:53:33 2012

Attributes:

b - black box (unknown)
 h - hierarchical
 n - noncombinational
 r - removable
 u - contains unmapped logic

Multibit Component : q0_reg

Cell	Reference	Library	Area	Width	Attributes
mid0/inst1/q0_reg[1]	**SEQGEN**		0.00	1	n, u
mid0/inst1/q0_reg[0]	**SEQGEN**		0.00	1	n, u
Total 2 cells			0.00	2	

Multibit Component : q0_reg

Cell	Reference	Library	Area	Width	Attributes
mid0/inst0/q0_reg[1]	**SEQGEN**		0.00	1	n, u
mid0/inst0/q0_reg[0]	**SEQGEN**		0.00	1	n, u
Total 2 cells			0.00	2	

Total 2 Multibit Components

SEE ALSO

create_multibit(2)
current_design(2)
remove_multibit(2)
report_cell(2)
report_compile_options(2)
report_reference(2)

report_multibit_banking

Reports all multibit registers in a design and the banking ratio.

SYNTAX

```
status report_multibit_banking  
  [-hierarchical]  
  [-nosplit]
```

ARGUMENTS

-hierarchical

Reports all multibit registers, the total number of sequential cells, and the banking ratio in each of the submodules across the design hierarchy.

-nosplit

Prevents line splitting. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

This command lists information about multibit registers in the design. The report includes the total number of single-bit and multibit sequential cells, the single-bit equivalent of these sequential cells, the sequential cell and flip-flop banking ratio, and the multibit register decomposition.

The sequential cells only include leaf cells. Clock-gating latches are excluded in this report. Therefore, the sequential cell count reported here can differ from what is reported by the **report_qor** and **report_area** commands.

The multibit register decomposition section reports the information about multibit cell references used in the design. See the example section for more details.

When you specify the **-hierarchical** option, the number of single-bit and multibit sequential cells and the sequential cell banking ratio is reported for each subdesign.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example generates a multibit banking report with the **-hierarchical** option:

```
prompt> report_multibit_banking -hierarchical
Total number of sequential cells:          2422
  Number of single-bit flip-flops:        1699
  Number of single-bit latches:           10
  Number of multi-bit flip-flops:         711
  Number of multi-bit latches:            2
Total number of single-bit equivalent sequential cells: 3147
  (A) Single-bit flip-flops:              1699
  (B) Single-bit latches:                  10
  (C) Multi-bit flip-flops:                1434
  (D) Multi-bit latches:                    4
Sequential cells banking ratio ((C + D) / (A + B + C + D)): 45.69%
Flip-Flop cells banking ratio ((C) / (A + C)): 45.77%
Multi-bit Register Decomposition:
-----
Bit-Width Reference      Number of instances  Single-bit Equivalent
-----
2-bits                   707 (29.19%)      1414 (44.93%)
  MB2_DFF1                705
  MB2_LD1                   2
4-bits                    6 (0.25%)       24 (0.76%)
  MB4_DFF1                  6
-----
Hierarchical multi-bit distribution:
-----
Hierarchical cell        Num. of MB  Num. of  Seq. cells
                        seq. cells seq. cells banking ratio
-----
top                       713    2422    45.69%
sub_design1               466    1055    61.28%
-----
1
```

The **Multi-bit Register Decomposition** section reports the information about multibit cell references used in the design. The information is reported for all bit widths of multibit references. In the above example, for 2-bit width multibit cells, the report is as shown:

```
-----
Bit-Width Reference      Number of instances  Single-bit Equivalent
-----
2-bits                   707 (29.19%)      1414 (44.93%)
  MB2_DFF1                705
  MB2_LD1                   2
-----
```

Here the number of 2-bit instances (707) corresponds to the total count of 2-bit multibit cell instances across all references (705 instances of flip-flop (MB2_DFF1) + 2 instances of latch (MB2_LD1)). The percentage of 2-bit instances (29.19%) is calculated as: $100 * \text{Number of 2-bit instances (707)} / \text{Total number of sequential cells (2422)}$. The number of single-bit equivalents (1414) is calculated as: $2 * \text{Number of 2-bit instances (707)}$. The corresponding single-bit equivalent percentage (44.93%) is calculated as: $100 * \text{Number of single-bit equivalents (1414)} / \text{Total number of single-bit equivalent sequential cells (3147)}$.

The **Hierarchical multi-bit distribution** section of the report is described in the following section:

The section reports the number of single-bit and multibit sequential cells and sequential cell banking ratio for each subdesign in the hierarchy.

Hierarchical cell	Num. of MB seq. cells	Num. of seq. cells	Seq. cells banking ratio
sub_design1	466	1055	61.28%

Here the **Num. of MB seq. cells** (466) is the total count of (2-bit and 4-bit) multibit flip-flops and latches in sub_design1. The **Num. of seq. cells** (1055) is the total count of single-bit and (2-bit and 4-bit) multibit flip-flops and latches in sub_design1. The **Sequential cell banking ratio** (61.28%) is the sequential cell (both latches and flip-flops) banking ratio in sub_design1.

SEE ALSO

create_multibit(2)
current_design(2)
report_cell(2)
report_compile_options(2)
report_reference(2)

report_mv_library_cells

Displays power management cells available in the target libraries.

SYNTAX

```
status report_mv_library_cells
  [-level_shifters]
  [-isolation_cells]
  [-retention_cells]
  [-switch_cells]
  [-always_on_cells]
  [-cell_name master_cell_name]
  [-verbose]
```

Data Types

master_cell_name string

ARGUMENTS

-level_shifters

Reports only library cells that have the level-shifter cell attribute **ls**.

-isolation_cells

Reports only library cells that have the isolation cell attribute **iso**.

-retention_cells

Reports only library cells that have the retention cell attribute **ret**.

-switch_cells

Reports only library cells that have the switch-cell attribute **switch_cg**.

-always_on_cells

Reports only library cells that have the always-on cell attribute **ao**.

-cell_name *master_cell_name*

Reports library information relative only to the given cell name.

-verbose

Reports all relevant multivoltage attributes cell by cell.

DESCRIPTION

This command reports all power management cells available in all target libraries. The output format is divided into two parts. The first is a library-based report, which prints all cells for each library available to the design. It is printed when the **-verbose** option is used.

The second is a cell-based summary, which is printed at all times. It prints the basic attributes of the cell and the library in which the cell is found.

Both reports are filtered by the available cell type options. The cell type options are not exclusive and can be combined in the command.

The report can also print one cell at the time by using **-cell_name** option. When the **-cell_name** option is used, all filtering type options are ignored.

EXAMPLES

The following example shows the output format in the default mode while reporting only isolation cells:

```
prompt> report_mv_library_cells -iso
```

```
*****
Report : report_mv_library_cells
Version: D-2010.03
Date   : Wed Aug 12 13:46:11 2009
*****
```

```
-----
Cell Based Report
-----
```

```
Cell: ISOHI
iso; Y=I+(!ISO); VSS(0.000,PRM); VDD(0.9,PRM); mylib1
iso; Y=I+(!ISO); VSS(0.000,PRM); VDD(1.1,PRM); mylib2
```

The following example shows the output format in verbose mode while reporting only isolation cells:

```
prompt> report_mv_library_cells -iso -verbose
```

```
*****
Report : report_mv_library_cells
Version: D-2010.03
Date   : Wed Aug 12 13:52:09 2009
*****
```

```
-----
Library Based Report
-----
```

```
Analyzing Target Library: mylib1 (PVT:1.0;0.9;0.0)
ISOLATION CELL: ISOHI
cell attribute:      iso;
logic function:      Z=I+(!ISO);
PG pins:             VSS(0.000,PRM); VDD(0.9,PRM);
```

related power supplies: I(VDD); ISO(VDD,iso); Z(VDD);
power down function:

Analyzing Target Library: mylib2 (PVT:1.0;1.1;0.0)
ISOLATION CELL: ISOHI
cell attribute: iso;
logic function: Z=I+(!ISO);
PG pins: VSS(0.000,PRM); VDD(1.1,PRM);
related power supplies: I(VDD); ISO(VDD,iso); Z(VDD);
power down function:

Analyzing Target Library: mylib3 (PVT:1.0;1.1;0.0)
Analyzing Target Library: mylib4 (PVT:1.0;0.9;0.0)

Cell Based Report

Cell: ISOHI
iso; Z=I+(!ISO); VSS(0.0,PRM); VDD(0.9,PRM); mylib1
iso; Z=I+(!ISO); VSS(0.0,PRM); VDD(1.1,PRM); mylib2

1

SEE ALSO

report_lib(2)

report_mv_qor

Estimates and reports static power for the power domains of a design in UPF mode. This command is supported only in DC Explorer.

SYNTAX

```
status report_mv_qor
  [-verbose]
  [-domains power_domain_list]
  [-output output_file_name]
  [-html]
```

Data Types

```
power_domain_list  list
output_file_name  string
```

ARGUMENTS

-verbose

Displays additional detailed information for the power domains. Each power domain is reported separately. It breaks down area, leakage power and number of cells between isolation, level shifter, always-on, macro, I/O pad and standard cells. It also provides a total of area, leakage, and cell count for power management cells (level shifters, isolation cells, and always-on cells).

-domains *power_domain_list*

Reports estimated leakage power, area, and number of cells for the specified power domains. Each power domain is reported separately if the **-verbose** option is specified. If you do not specify the **-domains** option, all the power domains are reported.

-output *output_file_name*

Specifies a single file into which the outputs are to be written. By default, the command writes the report into ASCII format with the specified file name. If **-html** option is specified together, it writes the specified file in HTML format.

-html

Reports in HTML format. If the **-output** option is not specified, the command prints a temporary file and launches a browser specified by the Tcl variable 'web_browser' to open the temporary file. If the **-output** option is specified, the command writes the reports to the specified file in HTML format.

DESCRIPTION

The **report_mv_qor** command estimates and reports static power for the list of power domains or all the power domains of a design. It also displays the primary voltage of the domain and its switching behavior if the **-verbose** option or the **-html** option is specified.

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following example shows a summary report of the **report_mv_qor** command. A very low-effort analysis is performed to estimate the design's static power values.

```
prompt> report_mv_qor
```

```
*****
Report : report_mv_qor
Design : ChipTop
Version: ...
Date   : ...
*****
```

Power Domain	Number of Instances	Area	Leakage power
TOP	1	0.00	0.0000 nW
MULT	3258	9420.13	1.1099 uW
GPRS	3328	11961.14	645.1089 nW
GENP	3604	8051.05	902.8282 nW

The following example shows the details for every power domain.

```
prompt> report_mv_qor -verbose
```

```
*****
Report : report_mv_qor
Design : ChipTop
Version: ...
Date   : ...
*****
```

Power Domain	Number of Instances	Area	Leakage power
TOP	1	0.00	0.0000 nW
MULT	3258	9420.13	1.1099 uW
GPRS	3328	11961.14	645.1089 nW
GENP	3604	8051.05	902.8282 nW

Power domain TOP

```
-----
Primary Voltage      : 1.08 v
```

Switching Behavior : Always on

Breakdown for power domain TOP

Category	Number of Instances	Area	Leakage power
Isolation Cells	0	0.00	0.0000 nW
Level Shifters	0	0.00	0.0000 nW
Always On Cells	0	0.00	0.0000 nW
Power management Cells	0	0.00	0.0000 nW
Macros	0	0.00	0.0000 nW
IO Pads	0	0.00	0.0000 nW
Standard Cells	1	0.00	0.0000 nW
Total	1	0.00	0.0000 nW

Power domain MULT

Primary Voltage : 1.08 v
Switching Behavior : Always on

Breakdown for power domain MULT

Category	Number of Instances	Area	Leakage power
Isolation Cells	0	0.00	0.0000 nW
Level Shifters	0	0.00	0.0000 nW
Always On Cells	0	0.00	0.0000 nW
Power management Cells	0	0.00	0.0000 nW
Macros	0	0.00	0.0000 nW
IO Pads	0	0.00	0.0000 nW
Standard Cells	3258	9420.13	1.1099 uW
Total	3258	9420.13	1.1099 uW

Power domain GPRS

Primary Voltage : 1.08 v
Switching Behavior : Shutdown

Breakdown for power domain GPRS

Category	Number of Instances	Area	Leakage power
Isolation Cells	0	0.00	0.0000 nW
Level Shifters	0	0.00	0.0000 nW
Always On Cells	0	0.00	0.0000 nW
Power management Cells	0	0.00	0.0000 nW
Macros	0	0.00	0.0000 nW
IO Pads	0	0.00	0.0000 nW
Standard Cells	3328	11961.14	645.1089 nW
Total	3328	11961.14	645.1089 nW

Power domain GENP

Primary Voltage : 1.08 v
Switching Behavior : Shutdown

Breakdown for power domain GENP

Category	Number of Instances	Area	Leakage power
Isolation Cells	0	0.00	0.0000 nW
Level Shifters	0	0.00	0.0000 nW
Always On Cells	0	0.00	0.0000 nW
Power management Cells	0	0.00	0.0000 nW
Macros	0	0.00	0.0000 nW
IO Pads	0	0.00	0.0000 nW
Standard Cells	3604	8051.05	902.8282 nW
Total	3604	8051.05	902.8282 nW

The following example displays a report for the specified power domains.

prompt> **report_mv_qor -domains {Multiplier/GENP MULT} -verbose**

Report : report_mv_qor

Design : ChipTop

Version: ...

Date : ...

Power Domain	Number of Instances	Area	Leakage power
GENP	3604	8051.05	902.8282 nW
MULT	3258	9420.13	1.1099 uW

Power domain GENP

Primary Voltage : 1.08 v

Switching Behavior : Shutdown

Breakdown for power domain GENP

Category	Number of Instances	Area	Leakage power
Isolation Cells	0	0.00	0
Level Shifters	0	0.00	0
Always On Cells	0	0.00	0
Power management Cells	0	0.00	0
Macros	0	0.00	0
IO Pads	0	0.00	0
Standard Cells	3604	8051.05	902.8282 nW
Total	3604	8051.05	902.8282 nW

Power domain MULT

Primary Voltage : 1.08 v

Switching Behavior : Always on

Breakdown for power domain MULT

Category	Number of Instances	Area	Leakage power
Isolation Cells	0	0.00	0

Level Shifters	0	0.00	0
Always On Cells	0	0.00	0
Power management Cells	0	0.00	0
Macros	0	0.00	0
IO Pads	0	0.00	0
Standard Cells	3258	9420.13	1.1099 uW
Total	3258	9420.13	1.1099 uW

SEE ALSO

- propagate_switching_activity(2)
- report_power_domain(2)
- set_power_prediction(2)
- set_switching_activity(2)
- power_keep_license_after_power_commands(3)

report_mw_lib

Displays information about the measurement units or reference libraries of a Milkyway library.

SYNTAX

```
status report_mw_lib
    [-unit_range]
    [-mw_reference_library]
    [mw_lib]
```

Data Types

mw_lib string

ARGUMENTS

-unit_range

Displays information of about the measurement units for the Milkyway library (length, time, capacitance, and so on) and the characteristics of the routing layers. This option requires you to specify the Milkyway library name.

-mw_reference_library

Reports a list of the reference libraries for the specified Milkyway design library, or for the current Milkyway library if no library name is specified in the command. You define the association between a design library and are reference library with the **set_mw_lib_reference** command.

mw_lib

Specifies the Milkyway library to be reported.

DESCRIPTION

This commands displays information about the measurement units or reference libraries of the current Milkyway library or a specified Milkyway library. Use the appropriate option, either **-unit_range** or **-mw_reference_library**, to specify the type of information to report.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example displays a list of reference libraries for the current Milkyway design library:

```
prompt> report_mw_lib -mw_reference_library
```

```
../ref/mw_lib/sc
../ref/mw_lib/io
../ref/mw_lib/ram8x64
../ref/mw_lib/ram16x128
1
```

The following example displays information about the measurement units and routing layers of a specified Milkyway design library:

```
prompt> report_mw_lib -unit_range my_mw_lib_A
```

Library: my_mw_lib_A

Tech. Attr.	Unit	Resolution	Min. Value	Max. Value
length	micron	1000	0.001	2147483.647
time	ns	1000	0.001	2147483.647
capacitance	pf	10000000	0.0000001	214.7483647
resistance	kohm	10000000	0.0000001	214.7483647

```
=====
length      micron      1000      0.001  2147483.647
time        ns          1000      0.001  2147483.647
capacitance pf          10000000  0.0000001  214.7483647
resistance  kohm         10000000  0.0000001  214.7483647
```

Layer: POLY

Mask name: poly

Attribute	Minimum	Maximum
thickness	0.0e+00	0.0e+00
unit resistance	0.0e+00	0.0e+00
unit capacitance	0.0e+00	0.0e+00
...		

```
=====
thickness      0.0e+00  0.0e+00
unit resistance 0.0e+00  0.0e+00
unit capacitance 0.0e+00  0.0e+00
...
```

```
1
```

SEE ALSO

close_mw_lib(2)

open_mw_lib(2)

set_mw_lib_reference(2)

report_name_rules

Reports the values of name rules.

SYNTAX

```
status report_name_rules
      [name_rules]
```

Data Types

```
name_rules  string
```

ARGUMENTS

name_rules

Specifies the name of the rules to be reported. If the *name_rules* option is not specified, all currently-defined name rules are reported.

DESCRIPTION

This command reports the values of a set of name rules. Name rules are created or modified by the **define_name_rules** command. "Special rules" reports the rule defined by the **-special** option. For details about **-special**, see the **define_name_rules** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **report_name_rules** command to report the values of the name rules called *EXAMPLE*:

```
prompt> report_name_rules EXAMPLE
```

```
*****
```

```
Report   : name_rules
Name Rules : EXAMPLE
```

Version : v3.0
 Date : Fri Sep 27 11:10:20 1991

Rules Name: EXAMPLE

Equal port and net names: false
 Collapse name space: false
 Case insensitive: true
 Special rules: none
 Reserved words: none

Max Repl Rem

Rules Type Len Char Chrs Prefix Allowed Chars

 Port Rules 16 '*' no P No restrictions
 Cell Rules 16 '*' no U No restrictions
 Net Rules 16 '*' no N No restrictions

The following example reports all currently-defined name rules. Three sets of name rules are defined as *EXAMPLE*, *MAPPING*, and *UPPER_ONLY*.

prompt> **report_name_rules**

 Report : name_rules
 Name Rules : All
 Version : v3.1
 Date : Thu Jan 21 11:03:57 1993

Rules Name: EXAMPLE

Equal port and net names: false
 Collapse name space: false
 Case insensitive: true
 Special rules: none
 Reserved words: none

Max Repl Rem

Rules Type Len Char Chrs Prefix Allowed Chars

 Port Rules 16 '*' no P No restrictions
 Cell Rules 16 '*' no U No restrictions
 Net Rules 16 '*' no N No restrictions

Rules Name: MAPPING

Equal port and net names: false
 Collapse name space: false
 Case insensitive: false
 Special rules: none
 Reserved words: none

Max Repl Rem

Rules Type Len Char Chrs Prefix Allowed Chars

 Port Rules none '_' no P No restrictions
 Mapping String: "{['_reg'],'reg'}"

Cell Rules none ' _ ' no U No restrictions
 Mapping String: "{{" _ reg","reg"}}"

Net Rules none ' _ ' no N No restrictions
 Mapping String: "{{" _ reg","reg"}}"

Rules Name: UPPER_ONLY
Equal port and net names: false
Collapse name space: true
Case insensitive: false
Special rules: none
Reserved words: none

 Max Repl Rem
Rules Type Len Char Chrs Prefix Allowed Chars

Port Rules none ' _ ' no P Use "A-Z _ "
Cell Rules none ' _ ' no U Use "A-Z _ "
Net Rules none ' _ ' no N Use "A-Z _ "

SEE ALSO

change_names(2)
define_name_rules(2)
report_names(2)

report_names

Reports potential name changes of ports, cells, and nets in a design.

SYNTAX

```
status report_names
  [-rules name_rules]
  [-hierarchy]
  [-dont_touch designs_list]
  [-nosplit]
  [-original]
```

Data Types

```
name_rules    string
designs_list  list
```

ARGUMENTS

-rules *name_rules*

Specifies a set of name rules to which names must conform. The *name_rules* must be defined with the **define_name_rules** command. By default, the name rules file specified in **default_name_rules** is used.

-hierarchy

Reports all names in the design hierarchy. By default, only objects within the current design are reported.

-dont_touch *designs_list*

Specifies a list of designs to which the **changes_names** command is not applied.

-nosplit

Prevents line splitting when column fields overflow.

-original

Reports original names only, and does not report changes.

DESCRIPTION

The **report_names** command reports the names of ports, cells, and nets that would be changed to conform to the specified name rules.

This command may not report all of the name changes that would occur by the **change_names** command.

The **report_names** is normally used before running the **change_names** command. The output of **report_names** can be redirected to a names file and used as input to **change_names**.

The *name_rules* file specifies the rules for modifying names, and must be defined with **define_name_rules**.

Use **report_name_rules** to display a list of name rules currently available. Refer to the **define_name_rules** man page for information about naming rules that can be affected when running **report_names**.

With no options specified, **report_names** operates on ports, cells, and nets in the current design. When **-hierarchy** is specified, the report is expanded to include all design objects within the current design object hierarchy.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports port, cell, and net names in the current design that do not conform to the rules defined in **default_name_rules**:

```
prompt> list default_name_rules
default_name_rules = "EXAMPLE"
```

```
prompt> report_names
```

```
*****
Report : names
       -rules EXAMPLE
Design : TOP
Version: v3.0
Date   : Tue Aug 13 14:24:23 1991
*****
```

Design	Type	Object	New Name
TOP	cell	U\$1	U_1
TOP	net	NET_NAME_IS_WAY_TOO_LONG	NET_NAME_IS_WAY_TOO
TOP	net	12345	N12345

```
prompt> change_names -names_file TOP.names
```

Information: 15 names changed using names file 'TOP.names'.

SEE ALSO

change_names(2)
define_name_rules(2)
report_name_rules(2)

report_net

Reports net information for the design of the current instance or for the current design.

SYNTAX

```
status report_net
  [-nosplit]
  [-noflat]
  [-transition_times]
  [-only_physical]
  [-verbose]
  [-cell_degradation]
  [-min]
  [-connections]
  [-physical]
  [net_list]
  [-significant_digits digits]
  [-max_toggle_rate]
  [-max_capacitance]
  [-scenarios scenario_list]
```

Data Types

```
net_list      list
digits        integer
scenario_list list
```

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line, starting in the correct column.

-noflat

Allows backward compatibility to produce net reports that only trace fanin and fanout for the current level of hierarchy in a design. The **report_net** command, by default, displays fanin and fanout information throughout the hierarchy for each net, as if the net were flattened. Earlier versions of **report_net** report the fanin and fanout for a net only at the current level of hierarchy.

-transition_times

Reports the rise and fall transition times for each net at the end of the report.

-only_physical

Reports the preroutes of the physical net.

-verbose

Reports verbose connection information when used with the **-connections** option.

When used with the **-physical** or **-only_physical** options, this option displays detailed physical information.

-cell_degradation

Reports the capacitance on the net, the limit for the capacitance on the net (calculated from the cell_degradation table for the drivers of the net), and violations on the net at the end of the report.

-min

Reports the minimum value of capacitance, resistance, or transition time instead of the maximum. By default, **report_net** displays only the maximum values.

-connections

Reports information about pins connected to the nets.

-physical

Reports the total wire length of the net and preroutes of the net, if preroute information exists.

net_list

Reports only those nets specified in *net_list*. If you do not specify *net_list*, all nets in the current instance are displayed.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are 0 through 13. The default value is 2. Using this option overrides the value set by the **report_default_significant_digits** variable.

-max_toggle_rate

Shows the maximum toggle rate values for each net.

-max_capacitance

Shows the maximum capacitance values for each net.

-scenarios *scenario_list*

Reports the timing derate for the specified scenarios in a multi-scenario design. Inactive scenarios are skipped in the report. Each scenario is reported separately. If you do not specify this option, only the current scenario is reported.

DESCRIPTION

The **report_net** command displays information about the nets in the design of the current instance or in the current design. If **current_instance** is set, the report is generated for the design of that instance. Otherwise, the report is generated for the current design.

The Load column is calculated as Load = Cell_input_cap + Wireload_cap. Attributes such as **dont_touch** and **annotated_capacitance** are displayed for each net. Note that **dont_touch** can be present on a net as an implicit attribute. This may

occur when the **set_dont_touch_network** command is used. All of the nets in the transitive fanout of a port are affected, but **dont_touch** cannot be removed independently from these nets. The annotated capacitance is the value set by the **set_load** command on a net.

If you specify the **-connections** option, the leaf cell pins connected to each net are listed.

Multicorner-Multimode Support

By default, this command uses information from the current scenario. Select different scenarios by using the **-scenarios** option.

EXAMPLES

The following is an example of a net report:

```
prompt> report_net
```

```
*****
Report : net
Design : CONTROL
Version: v3.1a
Date   : Fri 1993
*****
```

```
Operating Conditions: WCCOM
Wire Loading Model: 05x05
```

Attributes:

```
  d - dont_touch
  c - annotated capacitance
```

Net	Fanout	Fanin	Load	Pins	Attributes
ACCUMCL	1	1	0.39	2	
ACSEL0	1	1	0.39	2	
ACSEL1	1	1	0.39	2	
ACSEL2	1	1	0.39	2	d
BL0	0	2	0.39	2	
BL1	0	2	0.39	2	
BL2	0	2	0.39	2	
BL3	0	2	0.39	2	d
BOTCHAIN	1	1	1.39	2	
CHAININ	2	1	7.82	3	
CHAINOUT	1	1	0.39	2	
CLK	3	1	2.76	4	
DIN0	1	1	1.39	2	c
DIN2	1	1	0.00	2	
DIN3	1	1	0.00	2	
TOPCHAIN	1	1	1.39	2	
WEABLE	1	1	0.39	2	
ZEROIN	1	1	0.39	2	
n	2	2	2.32	4	
Total 50 nets	66	52	100.71	118	

Maximum	9	2	11.12	10
Average	1.32	1.04	2.01	2.36

The following example shows a net connection report for the net named t:

```
prompt> report_net -connections {t}

*****
Report : net
        -connections
Design : counter
Version: v3.1a
Date   : Fri 1993
*****
```

Connections for net 't':

Driver Pins	Type
-----	-----
t/Z	Output Pin (EO)
Load Pins	Type
-----	-----
t_bar/A	Input Pin (IVA)
zero/A	Input Pin (OR2)
a/C	Input Pin (AO2)

The following example shows a prerouted physical net report for the t net. It has one segment on metal m2 and a via at (900,281.70).

```
prompt> report_net -only_physical

*****
Report : net
Design : address
Version: 1999.10
Date   : Tue Apr 18 22:24:57 2000
*****

Net      Layer/Via  Special_Width Start_X Start_Y End_X End_Y
-----
t        5( m2)    10.00    202.40 -279.90 202.40 281.70
t        4(via45)   *    900.00 281.70   *    *
```

SEE ALSO

- current_design(2)
- current_instance(2)
- report_cell(2)
- report_constraint(2)
- report_design(2)
- report_internal_loads(2)
- set_dont_touch_network(2)


```
report_default_significant_digits(3)
```

report_net_fanout

Displays net fanout or buffer-tree information for the current design.

SYNTAX

```
status report_net_fanout
  [-nosplit]
  [-high_fanout]
  [-threshold lower]
  [-bound upper]
  [-verbose]
  [-connections]
  [-physical]
  [-min]
  [-tree [-depth level]]
  [net_list]
```

Data Types

<i>lower</i>	integer
<i>upper</i>	integer
<i>level</i>	integer
<i>net_list</i>	list

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line in the same column.

-high_fanout

Specifies to show high fanout nets only. A high fanout net is a net with more fanouts than **500**. A high fanout buffer tree is a buffer tree with more leaf loads than the specified value. The fanouts cross hierarchies.

This option is mutually exclusive with the **-threshold** option.

-threshold *lower*

Specifies to only show nets with more fanout than *lower*. The fanouts cross hierarchies.

This option is mutually exclusive with the **-high_fanout** option.

-bound *upper*

Specifies to only show nets with fanout less than or equal to *upper*. The fanouts cross hierarchies.

-verbose

Displays verbose information.

-connections

Displays information about pins connected to the nets.

-physical

Displays location information when applicable.

-min

Shows the minimum conditions. By default, only maximum conditions are shown.

-tree

Indicates to treat a buffer tree as transparent. The leaf loads of the buffer tree are treated as the fanouts of the net. Hierarchical boundaries are not considered as leaf loads.

This option is mutually exclusive with the *net_list* argument.

-depth *level*

Displays only buffer trees with more levels than *level*. This option can only be used with the **-tree** option.

net_list

Specifies to process only those nets on the *net_list*. If *net_list* is not specified, all nets in the current instance are processed.

This argument is mutually exclusive with the **-tree** option.

DESCRIPTION

The **report_net_fanout** command displays net fanout or buffer-tree information in the current design. If a *net_list* is specified, the report is generated only for the specified nets.

All reports cross hierarchical boundaries as if the nets were flattened. This is consistent with the **report_net** command.

The **report_net_fanout** command can be used to find high-fanout nets, buffer trees, or long buffer chains in the current design or the current instance.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to find high-fanout nets:

```
prompt> report_net_fanout -high_fanout
```

The following example shows how to find high-fanout nets in a particular set of nets:

```
prompt> report_net_fanout -high_fanout [get_nets scan*]
```

The following example shows buffer trees with 10 leaf pins:

```
prompt> report_net_fanout -tree -threshold 9 -bound 10
```

The following example shows how to find long buffer chains:

```
prompt> report_net_fanout -tree -depth 8 -bound 1
```

SEE ALSO

- `create_buffer_tree(2)`
- `current_design(2)`
- `current_instance(2)`
- `remove_buffer_tree(2)`
- `report_buffer_tree(2)`

report_net_routing_layer_constraints

Reports the routing layer constraints for the specified nets.

SYNTAX

```
status report_net_routing_layer_constraints  
  nets  
  [-output file_name]
```

Data Types

nets collection
file_name string

ARGUMENTS

nets

Specifies the nets for which to report routing layer constraints.

This is a required option.

-output *file_name*

Generates a Tcl script that contains the **set_net_routing_layer_constraints** commands used to set the routing layer constraints for the specified nets. You can use this file to set the same constraints in another session.

DESCRIPTION

This command reports the routing layer constraints for the specified nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the routing layer constraints for a net named netA.

```
prompt> report_net_routing_layer_constraints {netA}
```

SEE ALSO

```
remove_net_routing_layer_constraints(2)  
set_net_routing_layer_constraints(2)
```

report_net_routing_rules

Reports the nondefault routing rules for the specified nets.

SYNTAX

```
status report_net_routing_rules  
  nets  
  [-freeze]  
  [-minorchange]  
  [-normal]  
  [-output file_name]
```

Data Types

nets collection
file_name string

ARGUMENTS

nets

Specifies the nets for which to report nondefault routing rules.

This is a required argument.

-freeze

Reports the specified nets with the **freeze** attribute.

This option is not supported in Design Compiler topographical mode.

-minorchange

Reports the specified nets with the **minorchange** attribute.

This option is not supported in Design Compiler topographical mode.

-normal

Reports the specified nets with the **normal** attribute.

This option is not supported in Design Compiler topographical mode.

-output *file_name*

Generates a Tcl script with the specified file name that contains **set_net_routing_rule** commands for the current net routing rule settings of the specified nets. You can use this script to set the same net routing rules in another session.

DESCRIPTION

This command lists the nondefault routing rules for the specified nets. In addition, you can generate a Tcl script that contains **set_net_routing_rule** commands for the current net routing rule settings of the specified nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the nondefault routing rules for the nets named CLK1 and CLK2.

```
prompt> report_net_routing_rules {CLK1 CLK2}

*****
Report : net routing rules
Design : super_watch
Version: C-2009.06
Date   : Fri May 8 21:39:28 2009
*****

----- Nets with non-default routing rules -----
CLK1 : SP1
CLK2 : SP1

----- All other nets use DEFAULT routing rule -----
-----
```

The following example reports the nondefault routing rules for the nets named CLK3 and CLK4.

```
prompt> report_net_routing_rules {CLK3 CLK4}

*****
Report : net routing rules
Design : super_watch
Version: C-2009.06
Date   : Fri May 8 21:40:28 2009
*****

----- All nets use DEFAULT routing rule -----
-----
```

The following example reports the nondefault routing rules for all nets.

```
prompt> report_net_routing_rules [get_nets -hier]

*****
Report : net routing rules
Design : super_watch
```



```
Version: C-2009.06
Date  : Fri May 8 21:41:28 2009
*****
```

```
----- Nets with non-default routing rules -----
```

```
CLK1 : SP1
```

```
CLK2 : SP1
```

```
----- All other nets use DEFAULT routing rule -----
```

```
-----
```

The following example reports nets with the **freeze**, **minorchange**, and **normal** attributes.

```
prompt> report_net_routing_rules [get_nets io_c2*] \
-freeze -minorchange -normal
```

```
*****
```

```
Report : net routing rules
```

```
-freeze
```

```
-minorchange
```

```
-normal
```

```
Design : super_watch
```

```
Version: C-2009.06
```

```
Date  : Fri May 8 21:42:28 2009
```

```
*****
```

```
----- Nets with freeze attribute -----
```

```
io_c2[3]
```

```
io_c2[0]
```

```
----- Nets with minorchange attribute -----
```

```
io_c2[4]
```

```
io_c2[1]
```

```
----- Nets with normal attribute -----
```

```
io_c2[6]
```

```
io_c2[11]
```

```
io_c2[15]
```

```
io_c2[12]
```

```
io_c2[5]
```

```
io_c2[7]
```

```
io_c2[8]
```

```
io_c2[2]
```

```
io_c2[13]
```

```
io_c2[10]
```

```
io_c2[14]
```

```
io_c2[9]
```

SEE ALSO

set_net_routing_rule(2)

report_net_search_pattern

Reports the specified net search patterns.

SYNTAX

```
status report_net_search_pattern  
-pattern id | -all
```

Data Types

id integer

ARGUMENTS

-pattern *id*

Specifies the pattern ID that you want to report.

Reports the details of the search criteria used to define the specified net search pattern, such as its length, fanout, bounding box half perimeter, lower limits, and upper limits.

-all

Reports the details of all net search patterns defined for your design.

DESCRIPTION

The **report_net_search_pattern** command reports the settings of the specified net search patterns. You can specify a single net search pattern by using the **-pattern** option or all net search patterns by using the **-all** option.

Before you use this command, at least one net search pattern must exist in the design. To create a net search pattern, use the **create_net_search_pattern** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the details of the net search pattern with an ID of 42.

```
prompt> report_net_search_pattern -pattern 42
```

```
*****  
Report : net_search_pattern  
Design : CORE  
Version: F-2011.09-ICC-SP3  
Date   : Tue Nov  8 12:48:44 2011  
*****
```

```
pattern id: 42
```

```
-----  
fanout upper limit: 2  
bbox half perimeter lower limit: 135.785995  
bbox half perimeter upper limit: 162.809998  
length lower limit: 191.483994  
length upper limit: 221.309998  
blocked area ratio lower limit: 0.100000  
blocked area ratio upper limit: 0.250000  
region x1: 0.000000  
region y1: 0.000000  
region x2: 1195200.000000  
region y2: 1032840.000000  
aspect ratio lower limit: 0.003510  
aspect ratio upper limit: 0.049470  
-----
```

SEE ALSO

```
create_net_search_pattern(2)  
remove_net_search_pattern(2)  
set_net_search_pattern_delay_estimation_options(2)  
report_net_search_pattern_delay_estimation_options(2)  
set_net_search_pattern_priority(2)  
report_net_search_pattern_priority(2)  
get_matching_nets_for_pattern(2)
```

report_net_search_pattern_delay_estimation_options

Reports delay estimation options specified for a pattern or all net search patterns.

SYNTAX

status **report_net_search_pattern_delay_estimation_options**
-pattern *id* | -all

Data Types

id integer

ARGUMENTS

-pattern *id*

Specifies the pattern ID you want to query. When a valid ID is entered, any pattern-based delay estimation options assigned to that ID by the **set_net_search_pattern_delay_estimation_options** command are reported.

-all

Displays all pattern-based delay estimation options defined by the **set_net_search_pattern_delay_estimation_options** command.

DESCRIPTION

The **report_net_search_pattern_delay_estimation_options** command reports pattern delay estimation options for a net pattern or all net patterns that are defined.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The **report_net_search_pattern_delay_estimation_options** command can be used as shown in the following example:

```
prompt> report_net_search_pattern_delay_estimation_options -pattern 3
```

```
*****
Report : net_search_pattern_delay_estimation_options
Design : CORE
Version: F-2011.09-ICC-SP3
Date   : Tue Nov 8 12:48:44 2011
*****
```

Delay estimation options for pattern: 1

```
-----
minimum routing layer id: METAL1
maximum routing layer id: METAL2
-----
```

SEE ALSO

```
create_net_search_pattern(2)
report_net_search_pattern(2)
remove_net_search_pattern(2)
set_net_search_pattern_delay_estimation_options(2)
set_net_search_pattern_priority(2)
report_net_search_pattern_priority(2)
get_matching_nets_for_pattern(2)
```

report_net_search_pattern_priority

Reports the pattern matching priority.

SYNTAX

status **report_net_search_pattern_priority**

ARGUMENTS

DESCRIPTION

The **report_net_search_pattern_priority** command reports the current prioritization of all existing net patterns. If a priority list was not specified by the **set_net_search_pattern_priority** command, the **report_net_search_pattern_priority** command returns nothing. The default behavior of the **create_net_search_pattern** command is to prioritize patterns based on the order in which they were created. For example, if 5 net patterns were created, pattern 1 will have the highest matching priority, followed by pattern 2, and then pattern 3, and so on.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows a **report_net_search_pattern_priority** report:

```
prompt> report_net_search_pattern_priority
```

```
*****  
Report : net_search_pattern_priority  
Design : CORE  
Version: F-2011.09-ICC-SP3  
Date   : Tue Nov 8 12:48:44 2011  
*****
```

```
-----  
search priority is: 2 1  
-----
```

SEE ALSO

`create_net_search_pattern(2)`
`report_net_search_pattern(2)`
`remove_net_search_pattern(2)`
`set_net_search_pattern_delay_estimation_options(2)`
`report_net_search_pattern_delay_estimation_options(2)`
`set_net_search_pattern_priority(2)`
`get_matching_nets_for_pattern(2)`

report_ocvm

Displays information about advanced on-chip variation (AOCV) and parametric on-chip variation (POCV) derating tables and coefficients, and reports derating calculation details.

SYNTAX

```
status report_ocvm
  [-min]
  [-max]
  [-early]
  [-late]
  [-rise]
  [-fall]
  [-clock]
  [-data]
  [-cell_delay]
  [-net_delay]
  [-list_annotated]
  [-list_not_annotated]
  [-nosplit]
  [-arc_details]
  -type aocvm | pocvm
  [object_list]
```

Data Types

object_list list

ARGUMENTS

-min

Reports AOCV/POCV derate tables for the minimum operating condition.

-max

Reports AOCV/POCV derate tables for the maximum operating condition.

-early

Reports early AOCV/POCV derate tables.

-late

Reports late AOCV/POCV derate tables.

-rise

Reports rise AOCV/POCV derate tables.

-fall

Reports fall AOCV/POCV derate tables.

-clock

Reports clock AOCV/POCV derate tables.

-data

Reports data AOCV/POCV derate tables.

-cell_delay

Reports cell delay AOCV/POCV derate tables.

-net_delay

Reports net delay AOCV/POCV derate tables.

-list_annotated

Generates a list of leaf cells and global nets that are annotated with AOCV/POCV derate tables.

-list_not_annotated

Generates a list of leaf cells and global nets that are not annotated with AOCV/POCV derate tables.

-nosplit

Prevents split lines when columns overflow.

-arc_details

Reports the computed depth, distance and derate values of the timing arcs that start, pass through, or end at the leaf cells listed in *object_list*; or reports the POCV sigma tables annotated on the timing arcs of the library cells listed in *object_list*.

-type aocvm | pocvm

Specifies the OCV type. The allowed values are **aocvm**, which displays AOCV-related information, or **pocvm**, which displays POCV-related information.

object_list

Specifies the design objects to be reported, on which AOCV/POCV derate tables or coefficients have been annotated. Only leaf cells or library cells should be specified when this option is used with the **arc_details** option.

DESCRIPTION

This command reports user-specified information about advanced on-chip variation (AOCV) and parametric on-chip variation (POCV) analysis. The type of information displayed depends on the type of AOCV/POCV information previously annotated with the **read_aocvm** command.

If the OCV type is specified as **aocvm**, AOCV information is displayed. If the OCV type is specified as **pocvm**, POCV information is displayed.

Note: The Design Compiler tool supports only POCV derating, which is applied with the **read_ocvm** command. The IC Compiler tool supports both AOCV and POCV derating, which are applied by the **read_aocvm** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

In the following example, the design has been annotated with AOCV derate factors using the **read_aocvm** command:

```
prompt> report_ocvm -type aocvm
```

```
*****
```

```
Report : aocvm
```

```
Design : my_design
```

```
*****
```

	Fully	Partially	Not	
Total	annotated	annotated	annotated	
Leaf cells	6	0	4	2
Nets	7	0	0	7
	13	0	4	9

```
1
```

In the following example, the design has been annotated with POCV derate factors using the **read_aocvm** command:

```
prompt> report_ocvm -type pocvm
```

```
*****
```

```
Report : pocvm
```

```
Design : my_design
```

```
*****
```

POCV coefficient:

	Fully	Partially	Not	
Total	annotated	annotated	annotated	
Leaf cells	6	0	4	2
Nets	7	0	0	7
	13	0	4	9

POCV distance:

	Fully	Partially	Not	
Total	annotated	annotated	annotated	
Leaf cells	6	0	4	2
Nets	7	0	0	7
	13	0	4	9

```
1
```

The following example displays metrics and AOCV derate factors for a cell specified in the *object_list*. The arc depth displayed in the report is the most pessimistic depth for all possible paths through the arc.

prompt> **report_ocvm -type aocvm -arc_details [get_cells u4]**

Report : ocvm
 object_set
 Design : top

...

From pin: u4/A
 To pin: u4/Z
 Arc type: cell (clock network)

AOCVM arc metrics	Launch	Capture
Depth	10.50	10.50

AOCVM arc derates	Launch	Capture
Max early rise	0.9145	0.9145
Max early fall	0.9043	0.9043
Max late rise	1.0890	1.0890
Max late fall	1.0998	1.0998
Min early rise	0.9145	0.9145
Min early fall	0.9043	0.9043
Min late rise	1.0890	1.0890
Min late fall	1.0998	1.0998

From pin: CLK
 To pin: u4/A
 Arc type: net (clock network)

AOCVM arc metrics	Launch	Capture
Depth	5.00	5.00

AOCVM arc derates	Launch	Capture
Max early rise	0.9032	0.9032
Max early fall	0.8940	0.8940
Max late rise	1.1010	1.1010
Max late fall	1.1106	1.1106
Min early rise	0.9032	0.9032
Min early fall	0.8940	0.8940
Min late rise	1.1010	1.1010
Min late fall	1.1106	1.1106

From pin: u4/Z
 To pin: u5/A
 Arc type: net (clock network)

AOCVM arc metrics	Launch	Capture
Depth	5.00	5.00

AOCVM arc derates	Launch	Capture
Max early rise	0.9032	0.9032
Max early fall	0.8940	0.8940
Max late rise	1.1010	1.1010
Max late fall	1.1106	1.1106
Min early rise	0.9032	0.9032
Min early fall	0.8940	0.8940
Min late rise	1.1010	1.1010
Min late fall	1.1106	1.1106

1

SEE ALSO

read_ocvm(2)
remove_ocvm(2)
timing_pocvm_enable_analysis(3)

report_opcond_inference

Reports the strategy and settings for operating condition inference.

SYNTAX

```
status report_opcond_inference
      [-object_list cells]
```

Data Types

cells collection

ARGUMENTS

-object_list *cells*

Specifies the cells on which the strategies for operating condition inference to be reported. If cells are not specified, reports the settings for the top design.

DESCRIPTION

This command reports the strategies for the operating condition inference. If cells are not specified, reports the settings for the top design.

Multicorner-Multimode Support

This command applies to all scenarios.

EXAMPLES

The following example reports the settings for the operating condition inference for the hierarchical cell **subA**.

```
prompt> report_opcond_inference -object_list [get_cells {subA}]
```

```
*****
```

```
Report : report_opcond_inference
Design : des_chip
```

Version: F-2011.09-ICC
Date : Wed Apr 6 16:48:46 2011

Cell	Cell type	Level	Match_process_temperature

subA	macro	exact	false
	pad	exact	false
	switch	exact	false

SEE ALSO

set_opcond_inference(2)

report_operating_conditions

Displays a specific operating condition or all operating conditions in a library.

SYNTAX

```
status report_operating_conditions  
-library library_name  
[-name op_cond_name]
```

Data Types

```
library_name  string  
op_cond_name string
```

ARGUMENTS

-library *library_name*

Specifies the name of the library that stores the operating conditions.

-name *op_cond_name*

Specifies the name of the operating conditions.

DESCRIPTION

The **report_operating_conditions** command reports a specific operating condition or all operating conditions in the specified library.

To create a new operating conditions, use the **create_operating_conditions** command.

To set operating conditions on the current design, use the **set_operating_conditions** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports a specific operating condition called *BEST* in the library named *a_lib*:

```
prompt> report_operating_conditions -library a_lib -name BEST
```

The following example reports all of the operating conditions in the library named *IBM_CMOS5S6_SC*:

```
prompt> report_operating_conditions -library IBM_CMOS5S6_SC
```

SEE ALSO

`create_operating_conditions(2)`

`set_operating_conditions(2)`

report_optimize_dft_options

Reports options for physical design-for-test (DFT) optimization.

SYNTAX

status **report_optimize_dft_options**

ARGUMENTS

The **report_optimize_dft_options** command has no argument.

DESCRIPTION

This command reports the value of the physical DFT options available in command **set_optimize_dft_options**. The options are - **repartitioning_method** and **-single_dir_option**.

SEE ALSO

remove_optimize_dft_options(2)
set_optimize_dft_options(2)

report_partitions

Lists the hierarchical designs and their associated attributes and relative size (estimated in RTL).

SYNTAX

```
status report_partitions
[-nosplit]
```

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

This command displays the partition-related information, including estimated relative size (or cell area, for mapped designs), number of instantiations, and which instance is the master instance.

EXAMPLES

The following is an example of a partition report:

```
prompt> report_partitions
```

```
*****
Report   : partitions
Design   : vampire (unmapped)
Date     : Thu Feb 8 14:47:48 2001
*****
```

Partition attributes :

(*) - partition.
% - the percentage w.r.t total design area.

d - dont_touch.

m(n) - multiple instantiated design (number of instantiation).

Designs	Attributes

vampire(*)	15.6%
tap_block(*)	6.8%, m(7), u1
add(*)	1.8%, m(8), u0/u0
mult(*)	1.5%, m(8), u0/u1
first_tap(*)	6.9%
cascade	1.5%
accum(*)	4.0%

SEE ALSO

set_compile_partitions(2)

report_pass_data

Reports the data files that are available for a design created by Automated Chip Synthesis.

SYNTAX

```
status report_pass_data
  [-hierarchy]
  [-pass_list pass_list]
  [design]
```

Data Types

<i>pass_list</i>	list
<i>design</i>	string

ARGUMENTS

-hierarchy

Requests data for all compile partitions in the hierarchy of the specified design. Because compile partitions with a **dont_touch** attribute are not modified, these partitions are not included in the report. By default, this command returns data only for the specified design.

-pass_list *pass_list*

Specifies a list of pass names. Pass names that contain the * (asterisk) wildcard character suffix are acceptable. Default pass names will be *default_prefix**, in which *default_prefix* is defined by *acs_default_pass_name*. An error occurs if a pass name does not exist.

design

Specifies the design to for which to report the data. You can specify only one design. An error occurs if the specified design does not exist in memory. By default, if you do not specify a design, Automated Chip Synthesis uses the current design.

DESCRIPTION

The **report_pass_data** command locates all passes in *pass_list* directories under the Automated Chip Synthesis working directory (as specified by **acs_work_dir**), and reports on the data files present in each pass for the specified design. Specifically, the command checks for the following files:

- pass/db/pre_compile/*design*.db
- pass/db/post_compile/*design*.db

- `pass/scripts/design.autoscr`

The file suffixes shown above are the default file suffixes. You can modify these suffixes using the **acs_db_suffix** variable for .db files, and the **acs_constraint_file_suffix** variable for the script file.

EXAMPLES

Assume you must run two passes (pass0 and pass1) for the design named *my_design*. To check on the data available for this design, execute one of the following commands:

```
prompt> report_pass_data my_design -pass_list {pass0 pass1}
```

```
prompt> report_pass_data my_design -pass_list {pass*}
```

This command generates the following report:

```
+-----+-----+-----+-----+-----+
| Design | Pass | Pre-compile | Post-compile | Script |
|        |      | Database    | Database      |        |
+-----+-----+-----+-----+-----+
| my_design | pass0 | YES        | YES          | YES    |
+-----+-----+-----+-----+-----+
|          | pass1 | YES        | NO           | YES    |
+-----+-----+-----+-----+-----+
```

SEE ALSO

current_design(2)
 acs_compile_script_suffix(3)
 acs_db_suffix(3)
 acs_work_dir(3)

report_path_budget

Displays budgeting information about a design.

SYNTAX

```
status report_path_budget
  [-to to_list]
  [-from from_list]
  [-through through_list]
  [-nworst paths_per_endpoint]
  [-max_paths max_path_count]
  [-input_pins]
  [-nets]
  [-transition_time]
  [-significant_digits digits]
  [-nosplit]
  [-all]
  [-verbose]
```

Data Types

<i>to_list</i>	list
<i>from_list</i>	list
<i>through_list</i>	list
<i>paths_per_endpoint</i>	integer
<i>max_path_count</i>	integer
<i>digits</i>	integer

ARGUMENTS

-to *to_list*

Reports only the paths to pins, ports, or clocks specified by *to_list*. The default is to report the longest path to an output port, with the worst slack within each path group.

-from *from_list*

Reports only the paths from pins, ports, or clocks specified by *from_list*. The default is to report the longest path to an output port, with the worst slack within each path group.

-through *through_list*

Reports only the paths through pins specified by *through_list*. The default is to report the longest path to an output port, with the worst slack within each path group.

-nworst *paths_per_endpoint*

Specifies the number of paths per endpoint to report. The default is 1.

-max_paths *max_path_count*

Specifies the number of paths per path group to report. The default is 1.

-input_pins

Specifies that input pins are to be shown in the path report. The default is to show only output pins. This option also shows the delays of the nets connected to these pins.

-nets

Specifies that nets are to be shown in the path report. The default is not to show nets. To show the delay for the nets, use the **-input_pins** option.

-transition_time

Specifies that the net transition time for each driving pin is to be shown in the path report. The default is not to show the net transition time for each driving pin.

-significant_digits *digits*

Specifies the number of reported digits to be to the right of the decimal point in the report. Valid values are 0 to 13. The default value is 2. Using this option overrides the value set by the **report_default_significant_digits** variable.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-all

Reports all hierarchical pins. By default, the command reports only the budget number on the hierarchical pins of the cell that have been budgeted. When the **-all** option is specified, the command reports budget numbers on all hierarchical pins on this path.

-verbose

Reports the budget number on every cell along the path.

DESCRIPTION

This command provides a report of budgeting information for the current design. The delay number shown for each block or cell shows the amount of budget allocated for that block or cell in the path. The overall budget for the path is the sum of these budgets (delays).

EXAMPLES

The following example shows a budgeting report using only the default values:

```
prompt> report_path_budget -th [get_pins i_driver/OUT]
```

```
*****
```

```
Report : budget
```

```
-path full
```

```
-delay max
```

```
-max_paths 1
```

```
Design : test
```

```
Version: 2002.05
```

```
Date : Tue Mar 19 11:20:24 2002
```

```
*****
```

```
Attributes
```

```
-----
```

```
U User Specified Budgets
```

```
A Automatic Budgets
```

```
F Fixed delay
```

```
Operating Conditions:
```

```
Wire Load Model Mode: top
```

```
Startpoint: i_driver/rIN_reg
```

```
(rising edge-triggered flip-flop clocked by clk1)
```

```
Endpoint: i_receiver/OUT_reg
```

```
(rising edge-triggered flip-flop clocked by clk2)
```

```
Path Group: clk2
```

```
Path Type: ax
```

```
Des/Clust/Port Wire Load Model Library
```

```
-----
```

```
test tc6a120m2 cba_core
```

```
Point Budget Delay Slack Type
```

```
-----
```

```
clock clk1 (rise edge)
```

```
200.00 200.00 0.00 F
```

```
clock network delay 0.00 0.00 0.00 F
```

```
i_driver/rIN_reg/CLK
```

```
0.00 0.00 0.00 A
```

```
i_driver/rIN_reg/Q 400.00 837.75 -437.75 U
```

```
i_driver/OUT 400.00 136.27 263.73 U
```

```
i_receiver/IN 0.00 0.00 0.00 F
```

```
i_receiver/OUT_reg/D
```

```
800.00 1024.02 -224.02 U
```

```
-----
```

```
Total 1800.00 2198.04 -398.04
```

```
Required time 708.44 708.44
```

```
-----
```

```
Slack -1091.56 -1489.60
```

The following example shows a budgeting report that displays the budget number on every cell along the path:

```
prompt> report_path_budget -verbose -th [get_pins i_driver/OUT]
```

```
*****
```

```
Report : budget
```

```
-path full
```



```

-delay max
-max_paths 1
Design : test
Version: 2002.05-preprod
Date  : Tue Mar 19 11:20:24 2002
*****

```

Attributes

```

U  User Specified Budgets
A  Automatic Budgets
F  Fixed delay

```

```

Operating Conditions:
Wire Load Model Mode: top

```

```

Startpoint: i_driver/rIN_reg
            (rising edge-triggered flip-flop clocked by clk1)
Endpoint: i_receiver/OUT_reg
            (rising edge-triggered flip-flop clocked by clk2)
Path Group: clk2
Path Type: max

```

Des/Clust/Port	Wire Load Model	Library
test	tc6a120m2	cba_core

Point	Budget	Delay	Slack	Type

clock clk1 (rise edge)				
	200.00	200.00	0.00	F
clock network delay	0.00	0.00	0.00	F
i_driver/rIN_reg/CLK				
	0.00	0.00	0.00	A
i_driver/rIN_reg/Q	400.00	837.75	-437.75	U
i_driver/U1/A	0.00	0.00	0.00	F
i_driver/U1/Y	400.00	136.27	263.73	U
i_driver/OUT	0.00	0.00	0.00	F
i_receiver/IN	0.00	0.00	0.00	F
i_receiver/U1/A	0.00	0.00	0.00	F
i_receiver/U1/Y	100.00	118.36	-18.36	U
i_receiver/U2/A	0.00	0.00	0.00	F
i_receiver/U2/Y	100.00	141.71	-41.71	U
i_receiver/U3/A	0.00	0.00	0.00	F
i_receiver/U3/Y	100.00	118.97	-18.97	U
i_receiver/U4/A	0.00	0.00	0.00	F
i_receiver/U4/Y	100.00	141.83	-41.83	U
i_receiver/U5/A	0.00	0.00	0.00	F
i_receiver/U5/Y	100.00	118.98	-18.98	U
i_receiver/U6/A	0.00	0.00	0.00	F
i_receiver/U6/Y	100.00	141.84	-41.84	U
i_receiver/U7/A	0.00	0.00	0.00	F
i_receiver/U7/Y	100.00	118.98	-18.98	U
i_receiver/U8/A	0.00	0.00	0.00	F
i_receiver/U8/Y	100.00	123.34	-23.34	U
i_receiver/OUT_reg/D				

	0.00	0.00	0.00	F

Total	1800.00	2198.04	-398.04	
Required time	708.44	708.44		

Slack	-1091.56	-1489.60		

SEE ALSO

- dc_allocate_budgets(2)
- set_user_budget(2)
- report_default_significant_digits(3)

report_path_group

Reports information about path groups in the current design.

SYNTAX

```
status report_path_group  
  [-nosplit]  
  [-expanded]  
  [-scenarios scenario_list]
```

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most design information is listed in fixed-width columns. When the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-expanded

Expands the paths when reporting.

-scenarios *scenario_list*

Reports path groups for the specified scenarios of a multi-scenario design. Inactive scenarios are skipped in the report. Each scenario is reported separately.

If you do not specify this option, only the current scenario is reported.

DESCRIPTION

The **report_path_group** command produces a report showing information about path groups in the current design. The report includes path groups automatically created by the **create_clock** command and groups manually created with the **group_path** command. Path groups are used to affect the calculation of maximum delay cost during optimization. You can display the cost of each group using the **report_constraint** command.

To remove all path groups in the current design, use the **reset_design** command. To remove certain paths from their current groups, use **group_path -default**.

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenarios**

option.

EXAMPLES

The following example reports all timing attributes set on the design:

```
prompt> report_path_group

*****
Report : path_group
Design : counter
Version: 1998.01
Date   : Wed Jul 9 13:51:14 1997
*****

Group Name  Weight  Range
-----
**default** 1.00   0.00
CLK         1.00   0.00
BLUE        1.00   5.00

Paths:
From        Through    To          Group Name
-----
*           *          CLK          CLK
Red_reg[5]  *          Blue_reg[5]  BLUE
```

SEE ALSO

```
create_clock(2)
current_design(2)
group_path(2)
report_constraint(2)
reset_design(2)
```

report_physical_constraints

Reports the physical constraint settings for the current design. This command is supported only in topographical mode.

SYNTAX

```
status report_physical_constraints  
  [-no_site_row]  
  [-pre_route]
```

ARGUMENTS

-no_site_row

Specifies that site rows are not reported. By default, the site rows are reported block by block.

-pre_route

Specifies that preroutes are reported if they exist. By default, preroutes are not reported.

DESCRIPTION

The **report_physical_constraints** command reports the physical constraints settings that are applied to the design. The command reports the following physical constraints:

- die area
- placement area
- utilization
- aspect ratio
- rectilinear outline
- port side
- port location
- cell location
- placement blockage
- wiring keepouts
- voltage area
- site rows
- bounds
- preroutes (net shapes, user shapes, vias, via arrays, and via cells)
- physical-only cells

To report the physical constraints of preroutes, specify the **-pre_route** option.

Note that because of priority considerations, the physical constraints with respect to the same type of object are not all reported by the **report_physical_constraints** command. The following two rules help determine which constraints are in effect and are reported.

1. For the core area constraints set by the commands listed below, the placement area constraint has the highest priority and rectilinear outline has a higher priority than utilization and aspect ratio:

```
set_placement_area
set_rectilinear_outline
set_utilization
set_aspect_ratio
```

2. Between the two constraints for ports, port location has a higher priority than port side. For example, assume the following constraints are set:

```
set_placement_area -coordinate { 10 10 1000 1000 }
set_utilization 0.9
set_port_location Clk -coordinate {100 200}
set_port_side -side top {Clk Reset}
```

When you run the **report_physical_constraints** command, the report shows these constraints as follows:

```
Placement area      { 0.000 0.000 100.000 100.000 }
```

```
PORT LOCATION 1      LOCATION
Clk                { 100.000 200.000 }
```

```
PORT SIDE 1          SIDE
Reset                top
```

The utilization and the port side of the **Clk** port are omitted due to the constraint priority.

EXAMPLES

The following example shows the command usage:

```
prompt> report_physical_constraints
```

The following example shows a sample report for physical-only filler cells:

CELL2	LOCATION	ORIENTATION	FIXED
fill_1	{3100.000 700.000 }	N	Yes
fill_2	{3000.000 600.000 }	N	Yes

SEE ALSO

```
create_bounds(2)
create_net_shape(2)
create_user_shape(2)
create_placement_blockage(2)
create_site_row(2)
create_wiring_keepouts(2)
```

```
create_voltage_area(2)
extract_physical_constraints(2)
set_aspect_ratio(2)
set_cell_location(2)
set_placement_area(2)
set_port_location(2)
set_port_side(2)
set_utilization(2)
write_physical_constraints(2)
```

report_pin_map

Displays package pin-map information set by the **read_pin_map** command.

SYNTAX

status **report_pin_map**

ARGUMENTS

The **report_pin_map** command has no arguments.

DESCRIPTION

The **report_pin_map** command displays package pin-map information set by the **read_pin_map** command on the current design.

EXAMPLES

The following is an example of the **report_pin_map** command:

```
prompt> report_pin_map
```

```
*****  
Report : BSD Pin Map  
Design : M1  
Version: Z-2007.03  
Date   : Fri Dec 21 08:44:55 2006  
*****
```

```
Package Name   : M1  
File Name      : /designs/bss_ut4_1.map
```

Port Name	Pin
in[2]	6
in[1]	5
in[0]	1

clk	20
en	21
out	22
out1[2]	25
out1[1]	24
out1[0]	23
tck	28
tms	29
tdi	30
tdo	32
trst_n	33
dummy1[0]	34
dummy1[1]	35
dummy1[2]	36
dummy2[1]	37
dummy2[2]	38
dummy2[3]	39

SEE ALSO

read_pin_map(2)

report_pin_name_synonym

Reports pin-name synonym definitions.

SYNTAX

status **report_pin_name_synonym**
[-nosplit]

ARGUMENTS

-nosplit

Prevents line splitting. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_pin_name_synonym** command displays all of the currently defined pin name synonyms.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following command set shows the pin-name synonym settings and then uses the **report_pin_name_synonym** command to report pin-name synonyms:

```
prompt> set_pin_name_synonym SYN_CD CD
1

prompt> set_pin_name_synonym SYN_QN QN
1

prompt> set_pin_name_synonym -full_name \
mid1/bot1/LSR0P_at_bot/SYN_R mid1/bot1/LSR0P_at_bot/R
1
```

```
prompt> set_pin_name_synonym -full_name this/is/a/synonym \
mid1/FJK2SP_at_mid/TI
1
```

```
prompt> set_pin_name_synonym -full_name mid1/FJK2SP_at_mid/J \
mid2/bot2/LSR0P_at_bot/Q
1
```

```
prompt> report_pin_name_synonym
```

```
*****
Report : pin name synonym
Design : top
Version: X-2005.09
Date   : Wed Jun 29 10:29:04 2005
*****
```

Synonym	Pin Name	Type
-----	-----	-----
this/is/a/synonym	mid1/FJK2SP_at_mid/TI	full name
mid1/FJK2SP_at_mid/J	mid2/bot2/LSR0P_at_bot/Q	full name
mid1/bot1/LSR0P_at_bot/SYN_R	mid1/bot1/LSR0P_at_bot/R	full name
SYN_QN	QN	simple name
SYN_CD	CD	simple name
-----	-----	-----

```
1
```

The following example first removes all of the pin-name synonyms and then reports the updated information:

```
prompt> remove_pin_name_synonym -all
1
```

```
prompt> report_pin_name_synonym
```

```
*****
Report : pin name synonym
Design : top
Version: X-2005.09
Date   : Wed Jun 29 10:29:04 2005
*****
```

```
No pin name synonym
```

```
1
```

SEE ALSO

```
remove_pin_name_synonym(2)
set_pin_name_synonym(2)
```

report_pipeline_scan_data_configuration

Displays options specified by the **set_pipeline_scan_data_configuration** command.

SYNTAX

status **report_pipeline_scan_data_configuration**

ARGUMENTS

The **report_pipeline_scan_data_configuration** command has no arguments.

DESCRIPTION

This command displays options specified by the **set_pipeline_scan_data_configuration** command on the current design.

The command reports the following information:

- Head Pipeline Clock
 - Tail Pipeline Clock
 - Head Pipeline Stages
 - Tail Pipeline Stages
-

EXAMPLES

The following is an example of a pipeline scan data configuration report:

```
prompt> report_pipeline_scan_data_configuration
```

```
*****
```

```
Report : Pipeline Scan Data Configuration
```

Design : CORE
Version: D-2010.03-20100127
Date : Wed Jan 27 23:12:50 2010

Head Pipeline Clock	CLK1
Tail Pipeline Clock	CLK1
Head Pipeline Stages	3
Tail Pipeline Stages	5

1

SEE ALSO

current_test_mode(2)
set_pipeline_scan_data_configuration(2)
reset_pipeline_scan_data_configuration(2)

report_port

Displays information about ports of the current instance or the current design.

SYNTAX

```
status report_port
  [-drive]
  [-verbose]
  [-physical]
  [-only_physical]
  [-nosplit]
  [-significant_digits digits]
  [port_list]
```

Data Types

digits integer
port_list list

ARGUMENTS

-drive

Specifies that the port report is to include only the sections showing the drive capability of input and inout ports. By default, this option is off.

-verbose

Specifies that the port report is to include all port information. By default, only a summary section is shown that lists all ports and their direction.

-physical

Reports the physical location of the port. By default, this option is off. This option is not supported in Design Compiler topographical mode and will be ignored.

-only_physical

Reports the physical only ports. By default, this option is off. This option is not supported in Design Compiler topographical mode and will be ignored.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. By default, this option is off.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that the command reports. Using this option overrides the value set by the **report_default_significant_digits** variable. Allowed values are 0 through 13. The default is 2 for all fields except "Pin Load" and "Wire Load," which have a default of 4. Specifying a valid value for significant digits overwrites the existing default value.

port_list

Indicates that the port report is to include only the specified ports and cannot be used if the current instance is set. By default, all ports in the current instance or current design are listed.

DESCRIPTION

This command displays information about ports in the design of the current instance or the current design. If the current instance has been set, the report is generated for the design of that instance. Otherwise the report is generated for the current design.

By default, a brief report is produced that includes all ports in the design. If you use the **-verbose** option, all types of information are included. If you use the **-drive** option, only drive information for input and inout ports is shown. If you specify a *port_list* value, the command includes only those ports in the report.

Attributes pertaining to the **set_driving_cell** command that assign the driving cells to the ports are reported in the "Attrs" column of the driving cell section of the verbose report. The letter **D** indicates that the **-dont_scale** and **-max** option of the **set_driving_cell** command are used, and the letter **d** indicates that the **-dont_scale** and **-min** option are used, the letter **N** indicates that the **-no_design_rule** and **-max** option are used, the letter **n** indicates that the **-no_design_rule** and **-min** option are used.

If the current instance has been set, the report is generated for the design of that instance. Note that attributes on these ports have no effect unless the current design is changed to that subdesign.

The "Input Delay" and "Output Delay" portions of the report list the minimum rise, minimum fall, maximum rise, and maximum fall delays. They also list the clock to which the delay is relative. If the clock name is followed by (f), the delay is relative to the falling edge of the clock. Otherwise the delay is relative to the rising edge. If the clock name is followed by (l), input delay is considered as if coming from a level-sensitive latch or output delay is considered as if going to a level-sensitive latch. By default, the delay is relative to a rising edge-triggered device.

The "Max Tran" and "Min Tran" columns list the maximum and minimum rise and fall times set by the **set_transition** command to this port.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows a brief port report:

```
prompt> report_port
```

```
*****
```

```
Report : port
```

```
Design : counter
```

Version: v3.1

Date : Mon 1992

Port	Dir	Pin	Wire Load	Max Load	Max Trans	Max Cap	Connection Class	Attrs
A	in		0.0000	0.0000	--	--	--	
B	in		0.0000	0.0000	--	--	--	
C	in		0.0000	0.0000	--	--	--	
CL	in		0.0000	0.0000	--	--	--	
CLK	in		0.0000	0.0000	--	--	--	
D	in		0.0000	0.0000	--	--	--	
JOKE	in		0.0000	0.0000	--	--	--	
L	in		0.0000	0.0000	--	--	--	
P	in		0.0000	0.0000	--	--	--	
RESET	in		0.0000	0.0000	--	--	--	
T	in		0.0000	0.0000	--	--	--	
CO	out		2.0000	0.0000	--	--	--	
QA	out		2.0000	0.0000	--	--	--	
QB	out		2.0000	0.0000	--	--	--	
QC	out		2.0000	0.0000	--	--	--	
QD	out		2.0000	0.0000	--	--	--	

The following example shows a verbose port report:

```
prompt> report_port -verbose
```

Report : port

-verbose

Design : s24099

Version: v3.4a-development

Date : Thu Oct 12 11:15:22 1995

Port	Dir	Pin	Wire Load	Max Load	Max Trans	Max Cap	Connection Class	Attrs
in1	in		0.0000	0.0000	--	--	--	
in2	in		0.0000	0.0000	--	--	--	
in3	in		0.0000	0.0000	--	--	--	
in4	in		0.0000	0.0000	--	--	--	
sel1	in		0.0000	0.0000	--	--	--	
sel2	in		0.0000	0.0000	--	--	--	
out1	out		0.0000	0.0000	--	--	--	
out2	out		0.0000	0.0000	--	--	--	
b1	inout		0.0000	0.0000	--	--	--	
b2	inout		0.0000	0.0000	--	--	--	

Port	External Number	Fanout Points	Wireload Model
in1	0	--	
in2	0	--	
in3	0	--	

in1	0	--
in2	0	--
in3	0	--


```

in4      0    --
sel1     0    --
sel2     0    --
out1     0    --
out2     0    --
b1       0    --
b2       0    --

```

```

      Input Delay
      Min      Max   Related Max
Input Port Rise  Fall Rise  Fall Clock Fanout
-----
in1      --  --  --  --  --  --
in2      --  --  --  --  --  --
in3      --  --  --  --  --  --
in4      --  --  --  --  --  --
sel1     --  --  --  --  --  --
sel2     --  --  --  --  --  --
b1       --  --  --  --  --  --
b2       --  --  --  --  --  --

```

```

      Driving Cell
Input Port Rise      Fall      Mult Attrs
-----
in1      Isi_10k/AN2/Z  Isi_10k/AN2/Z  -- N

```

```

      Pin Drive      Min  Min  Min  Cell
Input Port Rise  Fall  Trans Cap  Fanout Deg
-----
in1      --  --  --  --  --  --
in2      --  --  --  --  --  --
in3      --  --  --  --  --  --
in4      --  --  --  --  --  --
sel1     --  --  --  --  --  --
sel2     --  --  --  --  --  --
b1       --  --  --  --  --  --
b2       --  --  --  --  --  --

```

```

      Output Delay
      Min      Max   Related Fanout
Output Port Rise  Fall Rise  Fall Clock Load
-----
out1      --  --  --  --  --  0.00
out2      --  --  --  --  --  0.00
b1        --  --  --  --  --  0.00
b2        --  --  --  --  --  0.00

```

1

The following is an example of a verbose port report specifying the **-significant_digits** option:

```
prompt> report_port -verbose -significant_digits 3
```

```
*****
```

```

Report : port
-verbose
-significant_digits 3

```

Design : char1

Version: V-2004.06

Date : Thu Dec 25 23:20:11 2003

Port	Dir	Pin Load	Wire Load	Max Load	Max Trans	Max Cap	Connection Class	Attrs
i1	in	0.000	0.000	--	1.524	--		
i2	in	0.000	0.000	2.139	--	--		
i3	in	0.000	0.000	--	--	--		
o1	out	0.000	0.000	--	--	--		
o2	out	0.000	0.000	--	--	--		

Port	External Number	Max Wireload Model	Min Wireload Model	Min Pin Load	Min Wire Load
i1	1	--	--	--	--
i2	1	--	--	--	--
i3	1	--	--	--	--

Input Port	Input Delay Min Rise	Input Delay Max Fall	Related Rise	Related Fall	Max Clock	Max Fanout
i1	3.436	3.436	6.436	6.436	--	--
i2	3.436	3.436	6.436	6.436	--	--
i3	3.515	3.617	6.515	6.617	--	--

Input Port	Max Drive Rise	Min Drive Fall	Resistance Rise	Resistance Fall	Max Max	Min Min	Min Cap	Cell Fanout	Deg
i1	--	--	--	--	--	0.548	--	--	
i2	--	--	--	--	--	--	--	--	
i3	--	--	--	--	--	--	--	--	

Input Port	Max Tran Rise	Min Tran Fall	Max Tran Rise	Min Tran Fall
i1	3.832	3.832	--	--
i2	6.000	1.850	3.000	1.250
i3	3.832	3.832	--	--

Output Port	Output Delay Min Rise	Output Delay Max Fall	Related Rise	Related Fall	Fanout Clock	Fanout Load
o1	3.061	3.061	3.061	3.061	--	0.000
o2	3.061	3.061	3.061	3.061	--	0.000

1

SEE ALSO

characterize(2)
current_design(2)
current_instance(2)
report_net(2)
set_cell_degradation(2)
set_connection_class(2)
set_drive(2)
set_driving_cell(2)
set_equal(2)
set_fanout_load(2)
set_input_delay(2)
set_load(2)
set_logic_one(2)
set_logic_zero(2)
set_max_capacitance(2)
set_max_fanout(2)
set_max_transition(2)
set_min_capacitance(2)
set_opposite(2)
set_output_delay(2)
set_unconnected(2)
set_wire_load_mode(2)
set_wire_load_model(2)
set_wire_load_min_block_size(2)
set_wire_load_selection_group(2)
report_default_significant_digits(3)

report_power

Calculates and reports dynamic and static power for the design or instance.

SYNTAX

```
status report_power
  [-net]
  [-cell]
  [-only cell_or_net_list]
  [-hierarchy]
  [-levels level_value]
  [-verbose]
  [-flat]
  [-exclude_boundary_nets]
  [-include_input_nets]
  [-analysis_effort low | medium | high]
  [-nworst number]
  [-sort_mode mode]
  [-histogram [-exclude_leq le_val
  | -exclude_geq ge_val]]
  [-nosplit]
  [-scenarios scenario_list]
  [-groups group_list]
```

Data Types

<i>cell_or_net_list</i>	object_list
<i>level_value</i>	integer
<i>number</i>	integer
<i>mode</i>	string
<i>le_val</i>	float
<i>ge_val</i>	float
<i>scenario_list</i>	list
<i>group_list</i>	list

ARGUMENTS

-net

Reports the power consumption of nets. Use the **-net** option alone, or use it with the **-cell** option. By default, only the design's summary power information is reported when neither option is specified.

-cell

Reports the power consumption of cells. If there are physical-only cells in a hierarchical module, the physical-only cells and the nonphysical-only cells are listed separately.

When you use the **-cell** option, some entries in the power report might not apply to certain cells. The column entry for such cells is annotated with N/A. Use the **-cell** option alone or with the **-net** option to report the cells and nets. By default, only the design's summary power information is reported when neither option is specified.

In Design Compiler topographical mode, when you specify the **-cell** option, the power report includes estimated clock tree power numbers, if you do not use the **-only** option and if the power prediction was turned on in the previous run of the **compile_ultra** command. This entry is marked *CLOCK_TREE_EST* in the cell column.

-only *cell_or_net_list*

Specifies a list of cells and nets to display with the **-net** or **-cell** options. With this option, only the cells and nets in the *cell_or_net_list* are listed in the power report. If both **-net** and **-only** options are specified, the *cell_or_net_list* must contain at least one net. Similarly, if both **-cell** and **-only** options are specified, the *cell_or_net_list* must contain at least one cell.

If the **-net**, **-cell**, and **-only** options are specified together, the *cell_or_net_list* must contain at least one net and one cell. Physical-only cells in a hierarchical module are not reported unless they are explicitly specified in the cell list as shown in the following example:

```
report_power -cell -only [get_cells hier_module/* -all]
```

-hierarchy

Specifies that the report be in a hierarchical format, with power information reported on a block-by-block basis. Use the **-levels** option to limit the number of levels of hierarchy shown in the report. The **-sort** and **-nworst** options are ignored for this report. The switching, internal, and leakage power numbers are reported for each hierarchical block. The hierarchy is shown through indentations.

In Design Compiler topographical mode, when you specify the **-hierarchy** option, the power report includes estimated clock-tree power numbers if power prediction was turned on in the last run of the **compile_ultra** command. This entry is marked *CLOCK_TREE_EST*.

-levels *level_value*

Specifies the number of levels of hierarchy to display, in the hierarchical report. You can specify a positive integer, greater than 1, with this option. Hierarchy levels deeper than those specified in this option are not shown in the hierarchical report. This option is only used for the hierarchical report and is ignored for all other types of reports.

-verbose

If the **-net** or **-cell** option is specified, it displays additional detailed information about the power information of the cells and nets. With the **report_power -verbose** specified, the power group summary displays the cell counts of the power groups.

-flat

Specifies that the power report traverse the hierarchy and report objects at all lower levels (assumes a flat design hierarchy). The default behavior is to report objects at only the current level of hierarchy. For cell report, if the **-flat** option is not specified, the power reported for a subdesign is the total power estimated for that subdesign, including all of its contents.

-exclude_boundary_nets

Specifies an option that is now obsolete.

-include_input_nets

Includes the switching power of primary input nets in the power report. The default is to exclude boundary input nets. This option affects the nets that are chosen to be displayed in the net-specific report as well as the value of the total switching power. This option does not affect the cell leakage and internal power values. When this option is specified, the sum of the power values of different power groups might be different with the summary report.

-analysis_effort low | medium | high

Provides a trade off between runtime and accuracy. The default is **low**. Specifying **low** effort results in the fastest runtime and the lowest accuracy of power estimates. Specifying **medium** or **high** effort results in a longer run with increased levels of accuracy. The analysis effort is considered only during the estimation of switching activity information, and therefore, has an effect only when the design is not fully annotated with switching activity.

-nworst number

Filters the report so that it displays only the highest *number* power objects. This option is valid only if either **-net** or **-cell** or both the options are specified.

-sort_mode mode

Determines the sorting mode for report order and **-nworst** selection. The valid sorting modes for the **-net** or **-cell** options are as follows:

-net option	-cell option
-----	-----
name	name
net_static_probability	cell_internal_power
net_switching_power	cell_leakage_power
net_toggle_rate	dynamic_power
total_net_load	

When using both **-net** and **-cell** and specifying a sorting mode, you must select a sorting mode that is valid for both options. The mode is used for both the cell and the net reports.

If you do not explicitly set the sorting mode, a default is chosen based on the mode of the **report_power** command:

Mode	Implicit default
-----	-----
-net	net_switching_power
-cell	cell_internal_power
-net -cell	dynamic_power

-histogram -exclude_leq le_val | -exclude_geq ge_val

Displays a histogram-style report showing the number of nets in each power range. The **-exclude_leq** and **-exclude_geq** arguments are used to exclude data values less than *le_val* or greater than *ge_val*, respectively. Useful for displaying the range and variation of power in the design. This option displays the histogram report only if either **-net** or **-cell** is specified.

-nosplit

Prevents line splitting and facilitates writing applications to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line, starting in the correct column.

-scenarios scenario_list

Reports power for the specified scenarios of a multicornner-multimode design. Each scenario is reported separately. Inactive scenarios are skipped in the report. If you do not specify this option, only the current scenario is reported.

-groups group_list

Reports power for the specified power groups. By default all instances are placed in only 1 power group, so the sum of the predefined groups power equals the total power consumption of the design. However, if the design has multi-driven nets, then the switching power reported in the group will be smaller than the number reported in summary, since it will be divided by the number of drivers for the multi-driven nets. The valid power groups of the *group_list* are: *io_pad*, *memory*, *black_box*, *clock_network*, *register*, *sequential*, *combinational*.

io_pad	cells defined as part of the pad_cell group in the library
memory	cells defined as part of the memory group in the library
black_box	cells with no functional description in the library
clock_network	cells in the clock network excluding io_pad cells
register	latches and flip-flops driven by the clock network excluding io_pads and black_boxes
sequential	latches and flip-flops clocked by signals other than those in the clock network
combinational	nonsequential cells with a functional description

DESCRIPTION

The **report_power** command calculates and reports power for a design. The command uses the user-annotated switching activity to calculate the net switching power, cell internal power, and cell leakage power, and displays the calculated values in a power report. The **report_power** command needs switching activity information on all design nets, and uses a switching activity propagation mechanism to estimate switching activity information on nonannotated design objects.

The options enable you to specify cells and nets for reporting. The default operation is to display the summary of power values for only the current design. If a current instance is specified, the **report_power** command displays the summary power values for that instance. The power of an instance is estimated in the context of the higher-level design, which means using the switching activity and load of the higher-level design.

The **-verbose** option reports more details of the power information. The **-flat**, **-exclude_boundary_nets**, **-nworst**, and **-sort_mode** options enable the filtering of objects that are selected by the **report_power** command. The **-sort_mode** option also affects the formatting of the power reports by modifying the order of nets and cells that are displayed by the **report_power** command.

The **-histogram** option causes additional sections to be displayed in the power reports. The power histogram classifies the nets or cells into groups of power values, allowing for easier visual analysis of the range of power values and of the distribution of the nets and cells across that range. The **-histogram** option enables the pruning of objects in the histogram by excluding values greater than or less than specified values.

Power analysis uses the current tool's mechanism to obtain the loads. For example, for synthesis in nontopographical mode without parasitic back-annotation, the tool uses wire load models; it uses back-annotated capacitance information when available, and so on.

The standalone **report_power** command does not update extraction information, so it is recommended to use **extract_rc** before using **report_power**.

When you run the **report_power** command it checks-out a Power Compiler license. If a license is not available, the command terminates with an error message. Otherwise, the command proceeds normally. At the completion of the command, the Power Compiler license is released. To keep the license checked-out after the completion of the **report_power** command, set the **power_keep_license_after_power_commands** variable to **true**.

In topographical mode, if power prediction is enabled, **report_power** reports the correlated power. The reported power numbers include the power consumption by design components as well as the predicted power of missing components, such as clock tree. The correlated power is not reported if any option of the **report_power** is used.

When the leakage power model is set to **channel_width**, using the **set_leakage_power_model** command, the total design power

reported by the **report_power** command includes the total weighted sum of the channel widths for the design.

Creation and use of scenarios is supported in Design Compiler Graphical.

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following example shows a summary report of the **report_power** command. A medium-effort analysis is performed to estimate the design's power values.

```
prompt> report_power -analysis_effort medium
```

```
*****
Report : power
       -analysis_effort low
Design : ChipTop
Version: F-2011.09
Date   : Thu Jun 30 10:12:49 2011
*****
```

Library(s) Used:

```
power_lib.db (File: /remote/libraries/power_lib.db)
```

```
Operating Conditions: WCCOM Library: power_lib
Wire Load Model Mode: segmented
```

Design	Wire Load Model	Library
ChipTop	ZeroWireload	power_lib.db
InstructionDecoder	ZeroWireload	power_lib.db
GeneralPurposeRegisters	ZeroWireload	power_lib.db

```
Global Operating Voltage = 0.99
Power-specific unit information :
  Voltage Units = 1V
  Capacitance Units = 1.000000pf
  Time Units = 1ns
  Dynamic Power Units = 1mW (derived from V,C,T units)
  Leakage Power Units = 1uW
```

```
Cell Internal Power = 9.4997 mW (81%)
Net Switching Power = 2.2525 mW (19%)
-----
```

```
Total Dynamic Power = 11.7522 mW (100%)
```

```
Cell Leakage Power = 7.7869 uW
```


	Internal	Switching	Leakage	Total	
Power Group	Power	Power	Power	Power	(%) Attrs
io_pad	0.0000	0.0000	0.0000	0.0000	(0.00%)
memory	0.0000	0.0000	0.0000	0.0000	(0.00%)
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)
clock_network	0.0000	0.0000	0.0000	0.0000	(0.00%)
register	6.4008	0.2053	1.5931	6.6076	(56.19%)
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)
combinational	3.0989	2.0472	6.1938	5.1524	(43.81%)
Total	9.4997 mW	2.2525 mW	7.7869 uW	11.7678 mW	

The following example shows a net power report sorted by the **net_switching_power** option and filtered to display only five nets with the highest switching power. A low-effort analysis is performed to estimate the design's power values.

```
prompt> report_power -net -flat -nworst 5
```

```
*****
```

```
Report : power
```

```
-net
```

```
-analysis_effort low
```

```
-nworst 5
```

```
-flat
```

```
-sort_mode net_switching_power
```

```
Design : ALARM_BLOCK
```

```
Version: v3.2a
```

```
Date : Sun Jun 19 15:45:26 1994
```

```
*****
```

```
Library(s) Used:
```

```
power_lib.db (File: /remote/libraries/power_lib.db)
```

```
Operating Conditions:
```

```
Wire Loading Model Mode: enclosed
```

```
Design      Wire Loading Model  Library
```

```
-----
ALARM_BLOCK      0.5K_TLM      power_lib.db
ALARM_STATE_MACHINE 0.5K_TLM      power_lib.db
ALARM_COUNTER     0.5K_TLM      power_lib.db
ALARM_COUNTER_DW01_inc_6_0
                  0.5K_TLM      power_lib.db
```

```
Global Operating Voltage = 4.75
```

```
Power-specific unit information :
```

```
Voltage Units = 1V
```

```
Capacitance Units = 50.029999ff
```

```
Time Units = 1ns
```

```
Dynamic Power Units = 10uW (derived from V,C,T units)
```

```
Leakage Power Units = 1nW
```

```

      Total  Static Toggle Switching
Net      Net Load Prob. Rate  Power  Attrs
-----
ACOUNT/CLK      20.467  0.500  0.1000  115.5149
```

```

ACOUNT/n493      23.193  0.985  0.0250  32.7255
ASM/n225         9.165  0.985  0.0250  12.9314
ACOUNT/HRS_OUT[3] 6.365  0.537  0.0303  10.8763
ACOUNT/HRS_OUT[2] 5.161  0.537  0.0303  8.8202
-----
Total (5 nets)           18.0868 uW

```

The following example shows a **report_power** summary report in topographical mode with power prediction on. Note the PWR-620 information message that mentions that it is the correlated power.

```

prompt> set_power_prediction
prompt> compile_ultra -incremental
prompt> report_power -analysis_effort medium

```

Information: Updating design information... (UID-85)
 Performing probabilistic propagation through design.

```

*****
Report : power
        -analysis_effort medium
Design : ALARM_BLOCK
Version: v3.2a
Date   : Sun Jun 19 15:45:24 1994
*****

```

Library(s) Used:

power_lib.db (File: /remote/libraries/power_lib.db)

Operating Conditions:
 Wire Loading Model Mode: enclosed

Design	Wire Loading Model	Library
ALARM_BLOCK	0.5K_TLM	power_lib.db
ALARM_STATE_MACHINE	0.5K_TLM	power_lib.db
ALARM_COUNTER	0.5K_TLM	power_lib.db
ALARM_COUNTER_DW01_inc_6_0	0.5K_TLM	power_lib.db

Global Operating Voltage = 4.75
 Power-specific unit information :
 Voltage Units = 1V
 Capacitance Units = 50.029999ff
 Time Units = 1ns
 Dynamic Power Units = 1mW (derived from V,C,T units)
 Leakage Power Units = 1nW

Information: Reporting correlated power. (PWR-620)

```

Cell Internal Power = 157.0383 nW (86%)
Net Switching Power = 26.1555 nW (14%)
-----
Total Dynamic Power  = 183.1938 nW (100%)

Cell Leakage Power   = 1.0225 mW

```

SEE ALSO

propagate_switching_activity(2)
set_power_prediction(2)
set_switching_activity(2)
extract_rc(2)
power_keep_license_after_power_commands(3)

report_power_calculation

Displays the calculation of the internal power for a pin, the leakage power for a cell, or the switching power for a net.

SYNTAX

```
status report_power_calculation
  pin_cell_or_net_list
  [-state_condition boolean_eq_of_pins | default | all]
  [-path_sources pin_name | default | all]
  [-rise]
  [-fall]
  [-verbose]
  [-nosplit]
```

Data Types

```
pin_cell_or_net_list  object_list
boolean_eq_of_pins    string
pin_name              string
```

ARGUMENTS

pin_cell_or_net_list

Specifies the objects on which the power calculation reports. All objects in the list must be of the same type. The type of power calculation depends on the object type, as follows:

- The tool reports the internal power calculation for a pin.
- The tool reports the leakage power calculation for a cell.
- The tool reports the switching power calculation for a net.

-state_condition *boolean_eq_of_pins* | **default** | **all**

Reports only the tables or polynomials with matching state conditions. This option is valid only for pin or cell type objects. Use **-state_condition default** to specify the default state condition. Use **-state_condition all** to consider all states for reporting.

-path_sources *pin_name* | **default** | **all**

Reports only the tables or polynomials with matching path sources. This option is valid only for pin type objects. Use **-path_sources default** to specify the default path condition (no path source). Use **-path_sources all** to specify that all path sources must be considered for reporting.

-rise

Limits the report of internal power calculation to only the rise power.

-fall

Limits the report of internal power calculation to only the fall power.

-verbose

Increases the amount of information that is reported for cells or pins. For cells, the leakage power calculation report is appended by an internal power calculation report for all pins and all possible states and path sources of the cells in the object list. For pins, all of the possible states and paths are printed (equivalent to specifying **-state_condition all -path_sources all**).

-nosplit

Prevents line splitting and facilitates writing tools to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_power_calculation** command provides detailed power calculation information for the specified pin, cell, or net. Use this information for debugging or verifying power data in a logic library. Before using this command to display details of internal or leakage power calculation, read the logic library file into the system using the **read_lib** command. This command automatically compiles the library and enables the **report_power_calculation** command for internal and leakage power. Otherwise, the built-in security mechanism terminates the command when issued for pin internal power or cell leakage power calculation. This restriction does not apply to net switching power calculation.

It is considered best practice to use the **report_power** command before issuing **report_power_calculation**. This ensures the propagation of any missing switching activity. If the design is not completely annotated with switching activity when **report_power_calculation** is executed, the missing activity is propagated with the same analysis effort that was used during the last issued **report_power** command. If no power analysis is performed prior to issuing the **report_power_calculation** command, a low analysis effort is used for switching activity propagation.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows the calculation of path dependent internal power from input pin *B* to output pin *Y*. The contributions for rise and fall internal power are shown separately.

```
prompt> report_power_calculation U4/Y -state_condition default -path_sources B
```

```
*****
```

```
Report : power_calculation
```

```
U4/Y
```

```
-state_condition default
```

```
-path_sources B
```

```
Design : rpc
```

```
Version: V-2004.06
```

Date : Thu Feb 5 12:10:44 2004

Library(s) Used:

example (File: /root/libraries/example.db)

Operating Conditions: typical Library: example
 Wire Load Model Mode: segmented

Design	Wire Load Model	Library
rpc	a	example

Global Operating Voltage = 0.9
 Power-specific unit information :
 Voltage Units = 1V
 Capacitance Units = 1.000000pf
 Time Units = 1ns
 Dynamic Power Units = 1mW (derived from V,C,T units)
 Leakage Power Units = 1nW

PD Pin Internal Power Calculation

cell: U4
 pin: Y
 path source: B

Rise internal energy per transition = 0.143423

library model: NLPM

table variables:

X = total_output_net_capacitance = 0.4

Y = input_net_transition = 0.3

relevant portion of lookup table:

(X) 0.1050	(X) 0.3550	
(Y) 0.0500	(Z) 0.1310	(Z) 0.1410
(Y) 0.4510	(Z) 0.1320	(Z) 0.1420

$Z = A + B \cdot X + C \cdot Y + D \cdot X \cdot Y$

A = 0.1267 B = 0.0400

C = 0.0025 D = 0.0000

Z = 0.143423

(no scaling since the library has no internal power k-factors)

SDPD Rise Pin Toggle Rate = 0.164

sdpd rise pin toggle rate = path weight * sd rise pin toggle rate

path weight = 1

sd rise pin toggle rate = 0.164 (estimated)

PD Rise Pin Internal Power = 0.0235214

pin internal power = internal energy * pin toggle rate

Fall internal energy per transition = 0.143423

library model: NLPM

table variables:

X = total_output_net_capacitance = 0.4

Y = input_net_transition = 0.3
 relevant portion of lookup table:
 (X) 0.1050 (X) 0.3550
 (Y) 0.0500 (Z) 0.1310 (Z) 0.1410
 (Y) 0.4510 (Z) 0.1320 (Z) 0.1420

$Z = A + B \cdot X + C \cdot Y + D \cdot X \cdot Y$
 A = 0.1267 B = 0.0400
 C = 0.0025 D = 0.0000

Z = 0.143423
 (no scaling since the library has no internal power k-factors)

SDPD Fall Pin Toggle Rate = 0.164
 sdpd fall pin toggle rate = path weight * sd fall pin toggle rate
 path weight = 1
 sd fall pin toggle rate = 0.164 (estimated)

PD Fall Pin Internal Power = 0.0235214
 pin internal power = internal energy * pin toggle rate

Total Pin Internal Power = 0.0470429

The following is an example of a state-dependent leakage-power calculation report. An SPPM leakage power model was used for this example.

prompt> **report_power_calculation U5 -state_condition "A B"**

***** Report : power_calculation U5 -state_condition A B Design : rpc Version: V-2004.06 Date :
 Thu Feb 5 12:00:48 2004 *****

Library(s) Used:

example (File: /root/libraries/example.db)

Operating Conditions: typical Library: example Wire Load Model Mode: segmented

Design Wire Load Model Library ----- rpc a example

Global Operating Voltage = 0.9 Power-specific unit information : Voltage Units = 1V Capacitance Units = 1.000000pf Time Units = 1ns Dynamic Power Units = 1mW (derived from V,C,T units) Leakage Power Units = 1nW

SD Cell Leakage Power Calculation cell: U5 state condition: A B

Leakage Power Value = 0.1097 library model: SPPM domain: D1 original polynomial: "0.0134+0.0002a+0.0742b+0.0006ab"
 variables: a = temperature = 40 b = voltage = 0.9 reduced value = 0.1097 (no scaling for SPPM leakage power)

State Probability = 0.18 (estimated)

SD Leakage Power = 0.01974 cell leakage power = leakage power value * state probability

The following report shows how switching power calculation is performed:

prompt> **report_power_calculation q**

 Report : power_calculation
 q
 Design : rpc

Version: V-2004.06
Date : Thu Feb 5 12:12:32 2004

Library(s) Used:

example (File: /root/libraries/example.db)

Operating Conditions: typical Library: example
Wire Load Model Mode: segmented

Design	Wire Load Model	Library
rpc	a	example

Global Operating Voltage = 0.9
Power-specific unit information :
Voltage Units = 1V
Capacitance Units = 1.000000pf
Time Units = 1ns
Dynamic Power Units = 1mW (derived from V,C,T units)
Leakage Power Units = 1nW

Net Switching Power Calculation

net: q
driver: U4/Y

Switching Energy Per Transition = 0.162
switching energy = $0.5 * \text{capacitance} * \text{voltage}^2$
total net capacitance = 0.4
voltage = 0.9

Net Toggle Rate = 0.8 (user annotated)

Switching power = 0.1296
net switching power = switching energy * net toggle rate

SEE ALSO

report_lib(2)
report_power(2)
set_switching_activity(2)

report_power_domain

Reports information about the specified power domain.

SYNTAX

```
status report_power_domain
    [power_domains]
    [-hierarchy]
    [-nosplit]
    [-verbose]
    [object_list]
```

Data Types

power_domains list or collection

ARGUMENTS

power_domains

Specifies the power domains to be reported. The *power_domain* argument can be a collection of power domains or name patterns.

If this argument is not specified, the command reports all power domains in the current scope.

You cannot use this argument with the *-hierarchy* option.

-hierarchy

Reports all power domains in the current scope and all of its descendant scopes.

You cannot use this option with the *power_domain* argument.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-verbose

Reports all functions of the supply sets associated with the power domain.

DESCRIPTION

The **report_power_domain** command reports detailed information about existing power domains. If you specify the power domains, only those power domains are reported; otherwise, all power domains in the current scope are reported.

The command fails if it does not find the specified power domain.

The following information is reported for each power domain:

- The hierarchical name relative to the current scope
- The current scope
- The elements of the current scope
- The supply nets available for use (including the supply nets that are domain supply nets, used for isolation or retention strategies, and connected to switches)
- The supply sets available for use (including the supply sets that are domain supply sets, used for isolation or retention strategies, and connected to switches)
- The primary power and ground net
- The default_isolation power and ground net. Default isolation supply is reported if implicit supply handles are enabled. If suppress_iss design attribute is set on a power domain, default isolation supply will not be reported for that power domain.
- The default_retention power and ground net. Default retention supply is reported if implicit supply handles are enabled. If suppress_iss design attribute is set on a power domain, default retention supply will not be reported for that power domain.
- The isolation strategies and the isolation power and ground nets. Isolation power and ground nets are reported same as specified through **-isolation_supply_set**, **-isolation_power_net** and **-isolation_ground_net** options. If none of **-isolation_supply_set**, **-isolation_power_net** and **-isolation_ground_net** options are specified, and isolation **-location** is specified as self, domain's default_isolation.power is reported as isolation power net and domain's default_isolation.ground is reported as isolation ground net. If the isolation power net is specified but the isolation ground net is not specified then domain's primary.ground will be used as the isolation ground. If the isolation ground net is specified but the isolation power net is not specified then domain's primary.power is used as the isolation power. If the isolation **-location** is specified as parent or fanout and no power and ground nets are specified, the command does not report power and ground nets since there might be more than one power and ground nets for a given isolation strategy if parent or fanout falls in different domains for different elements used in the strategy. Instead **(#)** is reported. Also, in case of **-no_isolation**, no power and ground nets are reported.
- The retention strategies and its retention power and ground nets Retention power and ground nets are reported same as specified through **-retention_supply_set**, **-retention_power_net** and **-retention_ground_net** options. If none of **-retention_supply_set**, **-retention_power_net** and **-retention_ground_net** options are specified, domain's default_retention.power is reported as retention power net and domain's default_retention.ground is reported as retention ground net.
- The power switches
- The input and output supply nets
- The maximum voltage of each supply net if it has been set by the **set_voltage** command

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example reports the power domains in the current scope when the supply set handles are OFF:

```
prompt> report_power_domain
```

```
*****
Report : power_domain
Design : top
Version: J-2014.09-SP4-VAL
Date   : Wed Apr 15 11:38:48 2015
*****
```

```
-----
```

```
Power Domain      : mid1/A
Current Scope     : top
Elements          : mid1/bot1, mid1/bot2, mid1
Available Supply Nets : A_VDD, A_VSS, R_VDD, R_VSS, ISO_VDD, ISO_VSS,
                        SN_A_1, SN_A_2, SN_A_3, SN_A_4, SN_A_5, SN_A_6,
                        SW_NET_1, SW_NET_2, SW_NET_3, SW_NET_4
Available Supply Sets :
```

```
Connections      -- Power --      -- Ground --
Primary:         mid1/A_VDD      mid1/A_VSS
(Max Op. Voltage) (1.08)         (1.08)
Isolation: ISO_STR_1  mid1/ISO_VDD (1.88)  mid1/ISO_VSS (1.88)
Retention: RET_STR_1  mid1/R_VDD (1.68)   mid1/R_VSS (1.68)
Retention: RET_STR_2  mid1/R_VDD_1 (1.55)  mid1/R_VSS_1 (1.55)
```

```
Switches         -- Input --      -- Output --
Switch: mid1/SW1  mid1/SW_NET_1      mid1/SW_NET_2
Switch: mid1/SW2  mid1/SW_NET_3      mid1/SW_NET_4
```

```
-----
```

1

The following example reports power and ground nets of different isolation strategies defined in the power domain bot:

```
prompt> set_isolation iso -domain bot \
-applies_to output \
```

```
prompt> set_isolation_control iso -domain bot \
-isolation_signal EN \
-isolation_sense low \
-location self \
```

```
prompt> set_isolation iso_parent -domain bot \
-elements inner/DI \
```

```
prompt> set_isolation_control iso_parent -domain bot \
-isolation_signal EN \
-isolation_sense low \
-location parent \
```

```
prompt> set_isolation no_iso -domain bot \
-elements inner/D5 \
-no_isolation \
```

```
prompt> report_power_domain bot
```

```
*****
```

```
Report : power_domain
```

```
Design : outer
```

```
Version: J-2014.09-SP5
```

```
Date : Thu Apr 16 15:00:05 2015
```

```
*****
```

```
-----
```

```
Power Domain      : bot
```

```
Current Scope     : outer
```

```
Elements          : inner
```

```
Available Supply Nets : top_sn, vss, iso, bot_sn
```

```
Available Supply Sets : top.primary, bot.primary, top.default_isolation,
bot.default_isolation, bot.default_retention
```

```
Connections      -- Power --      -- Ground --
```

```
Primary:         bot_sn          vss
```

```
(Max Op. Voltage) (1.08)          (0.00)
```

```
Default Isolation: bot.default_isolation.power bot.default_isolation.ground
```

```
(Max Op. Voltage) (1.08)          (0.00)
```

```
Default Retention: bot.default_retention.power bot.default_retention.ground
```

```
(Max Op. Voltage) (1.08)          (0.00)
```

```
Isolation: iso      bot.default_isolation.power (1.08) bot.default_isolation.ground (0.00)
```

```
Isolation: iso_parent (#)          (#)
```

```
Isolation: no_iso   -              -
```

```
Retention: ret      bot.default_retention.power (1.08) bot.default_retention.ground (0.00)
```

```
(#) Not reported because of location parent/fanout.
```

```
-----
```

```
1
```

SEE ALSO

```
create_power_domain(2)
```

```
remove_power_domain(2)
```

report_power_gating

Reports the power-gating style of retention registers in the design.

SYNTAX

```
status report_power_gating  
    [cell_or_design_list]  
    [-missing]  
    [-unconnected]
```

Data Types

cell_or_design_list list

ARGUMENTS

cell_or_design_list

Specifies a list of cells or designs in which the power-gating styles of cells are reported. The default is the current design.

-missing

Reports the sequential cells that do not have the power-gating style.

-unconnected

Reports only the retention registers that are not stitched.

DESCRIPTION

This command reports the power-gating style of retention registers. The retention registers in the libraries have a cell-level **power_gating_cell** attribute to specify the power-gating styles of the registers. The command only reports the cells having the **power_gating_style** attributes set by the **set_power_gating_style** command. The command requires a Power Compiler license to run.

If **-missing** is specified, the command only reports the sequential cells that do not have the power-gating style.

If **-unconnected** is specified, the command only reports the retention registers whose power gating pins have not yet been stitched.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the power-gating styles of retention registers in *my_design*:

```
prompt>report_power_gating my_design
```

SEE ALSO

`power_enable_power_gating(3)`

report_power_model

Report information of power models in the design.

SYNTAX

```
status report_power_model  
[-verbose]
```

ARGUMENTS

-verbose

Report detailed information about power models in the design.

DESCRIPTION

This command reports power model information. It includes information about power models created in the `define_power_model` command and applied in the `apply_power_model` command.

In verbose mode, it also displays detailed information about the options used in the `define_power_model` and `apply_power_model` commands.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports all power models found in the design:

```
prompt> report_power_model  
  
*****  
Report : report_power_model  
Design : top  
*****
```

Column Header:

Elmt - Num of cells in apply_power_model -elements

SMap - Num of mappings in apply_power_model -supply_map

PMap - Num of mappings in apply_power_model -port_map

Cell - Num of cells applied with power model

Power Model	Defined In	Applied In	Elmt	SMap	PMap	Cell
model_1	User Library	macro_0	1	0	0	1
model_2	User Library	top	1	2	2	1
model_2	User Library	top	0	0	0	1
model_1	User Library	top	0	0	0	1

SEE ALSO

define_power_model(2)

apply_power_model(2)

report_power_pin_info

Reports the power pin information for leaf cells in the current design.

SYNTAX

```
status report_power_pin_info  
      cell_instances  
      [-nosplit]
```

Data Types

cell_instances collection

ARGUMENTS

cell_instances

Specifies the leaf cells for which to report the power pin information.

This is a required argument.

DESCRIPTION

The **report_power_pin_info** command reports the power pin information for leaf cells in the current design.

The command reports the name, types, voltage specification of the power pins, as well as the supply (power and ground) nets that are connected to the power pins.

If a supply connection is marked with an asterisk (*), it is an exception (explicit) connection. Exception supply connections are connections that are established with the **connect_supply_net** command.

If a supply connection is marked with an abbreviation drv (drv), it is a derived always-on (ao) connection for the buffer cell. The connection will be considered as derived if the buffer cell comes on the always-on path but is not an always-on buffer. Hence the connection is not made to its PG pins and **connect_supply_net** command is not seen in the UPF file written out by the tool. The buffer cells with derived connections are also reported by MV-076 message during **check_mv_design-connection_rules -verbose** command.

Exception supply connections can occur in the following situations:

- Explicit specification with the **connect_supply_net** command
- Power intent specification in the UPF file

- Level-shifter cell insertion during compilation

In any of these cases you can observe the exception connections by using the **write_script** or **save_upf** command. The connections appear as **connect_supply_net** commands in the generated files.

The best-case and worst-case voltages of the connected power net are reported, along with the name of the power net. If a voltage is not specified on the connected net, a hyphen (-) is printed in the respective voltage fields.

Hierarchical cells are ignored by this command. The tool issues the message "No power pins to report".

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports power pin information for the leaf cell instance named I0:

```
prompt> report_power_pin_info I0
```

```
*****
Report : power_pin_info
Design : top
Version: B-2008.09
Date   : Mon Mar 24 02:04:49 2008
*****
```

Note: Power connections marked by (*) are exceptional

Cell	Power Pin Name	Type	Power			Connected
			Best Case Voltage	Worst Case Voltage	Net	
I0	PWR	Primary Power	-	-	-	
I0	GND	Primary Ground	-	-	-	

1

The following example shows the voltage on the connected power net set:

```
prompt> report_power_pin_info [get_cells * -hier]
```

```
*****
Report : power_pin_info
Design : top
Version: B-2008.09
Date   : Mon Mar 24 02:04:49 2008
*****
```

Note: Power connections marked by (*) are exceptional

Cell	Power Pin	Type	Best Case	Worst Case	Connected Power

Name	Voltage	Voltage	Net

PD0_INST/I0 PWR	Primary Power	1.08	0.00 PD0_VDD
PD0_INST/I0 GND	Primary Ground	0.00	0.00 PD0_VSS
PD0_INST/I1 PWR	Primary Power	1.08	0.00 PD0_VDD
PD0_INST/I1 GND	Primary Ground	0.00	0.00 PD0_VSS
I0 PWR	Primary Power	1.08	0.00 PD0 (*)
I0 GND	Primary Ground	-	- -
I1 PWR	Primary Power	1.08	0.00 PD0 (*)
I1 GND	Primary Ground	-	- -

1			

The following example reports power pin information the buffer cell U1225 which comes on always-on path but its library cell is not marked as always-on in the technology library. Hence its supply connections are marked as (drv). Subsequent **check_mv_design** report shows MV-076 report for the cell.

prompt> **report_power_pin_info**

```
*****
Report : power_pin_info
Design : btest
Version: G-2012.06-SP1
Date   : Thu Aug 9 04:00:03 2012
*****
```

Note: Power connections marked by (*) are exception power connections
Note: Power connections marked by (drv) are derived as power connections

Cell	Power Pin Name	Type	Best Case Voltage	Worst Case Voltage	Connected Power Net

U1225	VDD	Primary Power	0.90	0.90	vdd (drv)
U1225	VSS	Primary Ground	0.00	0.00	vss (drv)

1					

prompt> **check_mv_design -connections_rules -verbose**

Always On Checks

Warning: Always on net 'n771' is driven by normal cell 'U1225' of type 'BUF_2'. (MV-076)

Always On Checks Summary

Warning: Found '1' always on nets that are driven by normal cells. (MV-080)

Please review report above for warnings and errors.

1

In the following example, a hierarchical cell is passed as an argument:

prompt> **report_power_pin_info [get_cells PD0_INST]**

```
*****
```

Report : power_pin_info
Design : top
Version: B-2008.09
Date : Mon Mar 24 02:04:49 2008

No power pins to report.
1

SEE ALSO

connect_supply_net(2)

report_power_switch

Reports all of the specified power switches.

SYNTAX

```
status report_power_switch  
    [power_switch_name]  
    [-verbose]
```

Data Types

power_switch_name string or list

ARGUMENTS

power_switch_name

Specifies a list of power switch names. The individual names must be hierarchical relative to the current scope. If the switch exists in the current scope, its simple name can be used. If no power switches specified absent, the tool reports all of the power switches in the current scope.

-verbose

Reports details about the specified power switches.

DESCRIPTION

The **report_power_switch** command enables you to get the information of the specified power switches. The power switch information reported includes its relative hierarchical name that also contains the scope in which it is created, the current scope name, the name of the simple power domain of which the switch is a part, its output supply net, and its input supply net.

The command also prints out a verbose report if the **-verbose** option is used. The verbose report contains all of the information specified above along with the ctrl_port names, the ack_port names, and the on and off state names with their respective Boolean functions.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports information about all power switches within the power domain named *PD1* in the current scope:

```
prompt> report_power_switch SWA
```

```
-----
Power Switch      : SWA
Current Scope     : mid1
Switch Power Domain : A
Input port net    : SNA_2
Output port net   : SNA_1
-----
```

```
1
```

The following example reports the information about the same power switches as above, this time using the **-verbose** option to show more detail:

```
prompt> report_power_switch SWA -verbose
```

```
-----
Power Switch      : SWA
Current Scope     : mid1
Switch Power Domain : A
Input port net    : SNA_2
Output port net   : SNA_1
Control Ports     : ctrl
Acknowledge Ports : ackport1, ackport2

On States        : <on1, ctrl>, <on2, !logic1 & ctrl>,
                  <on3, !logic1 & !ctrl>, <on4, !logic1 | ctrl>,
                  <on5, !logic1 | !ctrl>
Off States       : <off1, output>, <off2, output>
-----
```

```
1
```

SEE ALSO

create_power_switch(2)

report_preferred_routing_direction

Reports the preferred routing direction for all routing layers.

SYNTAX

string **report_preferred_routing_direction**

ARGUMENTS

The **report_preferred_routing_direction** command has no arguments.

DESCRIPTION

The **report_preferred_routing_direction** command reports the preferred routing direction for all layers from in-memory information.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of a preferred routing direction report:

```
prompt> report_preferred_routing_direction
```

```
*****  
Report : Layers  
Design : core_chip  
Version: Y-2006.06  
Date   : Thu Mar 23 03:43:06 2006  
*****
```

Layer Name	Library	Design	Tool understands
metal1	Horizontal	Not Set	Horizontal
metal2	Vertical	Not Set	Vertical
metal3	Horizontal	Not Set	Horizontal

metal4	Vertical	Not Set	Vertical
metal5	Horizontal	Vertical	Vertical
metal6	Vertical	Vertical	Vertical

WARNING: Consecutive layers have the same routing direction.
1

SEE ALSO

remove_preferred_routing_direction(2)
set_preferred_routing_direction(2)

report_pst

Reports the power states in the current design previously created with the **add_pst_state** command.

SYNTAX

```
status report_pst
  [-supplies supply_list]
  [-pst_state_limit number]
  [-verbose]
  [-scope instance_name]
  [-derived]
```

Data Types

```
supply_list    list
number         integer
instance_name string
```

ARGUMENTS

-supplies *supply_list*

Specifies a list of supply net or port names to be included in the report. The order of the names in this list determines their order in the report. By default, the report includes all power nets in the current design.

-pst_state_limit *number*

Specifies a limit on the number of states reported in the derived power state table, an integer between 1 and 1,000,000. The default is 100,000. The derived power state table is not reported if the number of states is over the limit. The limit applies whenever the derived power state table is reported, including when the **-supplies** option is used, or the current scope is a lower scope and the **-derived** option is not used.

-verbose

Expands the report to explicitly list all the power states. By default, the report is shortened by showing only an asterisk (*) when all states of a net or port are used.

-scope *instance_name*

Specifies a design instance different from current design. The resulting power state table contains only the supply states derived from all power state tables contained in the specified design instance.

-derived

Reports the power state table at the top level, even if the current scope is a lower-level scope.

DESCRIPTION

This command reports the power state table of the current design. This table is the result of combining all user-defined power state tables in the current design and its logical hierarchy.

You create power state tables by using the **add_port_state**, **create_pst**, and **add_pst_state** commands.

If no power state table has been explicitly created with the **create_pst** command, the derived power state table is full, meaning that all combinations of power states are allowed. Just after a new power state table is explicitly created, the table is empty; you create new power states and add them to the table by using the **add_pst_state** command.

Note: You can also display the power state table in the Design Vision GUI.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following UPF script creates a power state table, and the **report_pst** command reports the resulting table.

```
## POWER STATE TABLE
add_port_state VDD -state {TOP 1.08}
add_port_state I_ALU/VDDT -state {BLOCK_1 0.7}
add_port_state I_STACK_TOP/I_STACK_TOP_sw/out \
  -state {BLOCK 1.08} -state {BLOCK_off off}
add_port_state I_REG_FILE/I_REG_FILE_sw/out \
  -state {BLOCK 1.08} -state {BLOCK_off off}
add_port_state VSS -state {GND 0.0}
create_pst risc_core_pst \
  -supplies {VDD I_ALU/VDDT I_STACK_TOP/VDDTSI_REG_FILE/VDDTS VSS}
add_pst_state s0 -pst risc_core_pst -state {TOP BLOCK_1 BLOCK BLOCK GND}
add_pst_state s1 -pst risc_core_pst -state {TOP BLOCK_1 BLOCK_off BLOCK GND}
add_pst_state s2 -pst risc_core_pst -state {TOP BLOCK_1 BLOCK BLOCK_off GND}

prompt> report_pst
...
Scope: RISC_CORE
Supply Names: I_ALU/VDDT, I_ALU/VDD, I_ALU/VSS, I_STACK_TOP/I_STACK_TOP_sw/out,
  I_REG_FILE/I_REG_FILE_sw/out,
Resulting Power States in this Scope:
  I_ALU/VDDT|
    I_ALU/VDD|
      I_ALU/VSS|
        I_STACK_TOP/I_STACK_TOP_sw/out|
          I_REG_FILE/I_REG_FILE_sw/out|
drv:      *| *| *| BLOCK| *|
drv:      *| *| *| *| BLOCK|

User PST Name: risc_core_pst
Defined in Scope: RISC_CORE
```

```

Total Power States: 3
  VDD|
    I_ALU/VDDT|
      I_STACK_TOP/VDDTS|
        I_REG_FILE/VDDTS|
          VSS|
s0:  TOP| BLOCK_1|  BLOCK|  BLOCK|  GND|
s1:  TOP| BLOCK_1| BLOCK_off| BLOCK|  GND|
s2:  TOP| BLOCK_1|  BLOCK| BLOCK_off| GND|

```

The notation "drv:" in the report indicates that the states were derived from the list of possible port states are were not explicitly defined in an **add_port_state** command.

The asterisks in the report indicate where all states of a supply can be used. To expand the asterisks to show the full list of state names, use the **-verbose** option. For example,

```

prompt> report_pst -verbose
...
Scope: RISC_CORE
Supply Names: I_ALU/VDDT, I_ALU/VDD, I_ALU/VSS, I_STACK_TOP/I_STACK_TOP_sw/out,
              I_REG_FILE/I_REG_FILE_sw/out,
Resulting Power States in this Scope:
  I_ALU/VDDT|
    I_ALU/VDD|
      I_ALU/VSS|
        I_STACK_TOP/I_STACK_TOP_sw/out|
          I_REG_FILE/I_REG_FILE_sw/out|
drv:  BLOCK_1|  TOP|  GND|  BLOCK|  BLOCK|
drv:  BLOCK_1|  TOP|  GND|  BLOCK| BLOCK_off|
drv:  BLOCK_1|  TOP|  GND| BLOCK_off|  BLOCK|
...

```

SEE ALSO

```

add_port_state(2)
add_pst_state(2)
create_pst(2)
set_scope(2)

```

report_qor

Displays QoR information and statistics for the current design.

SYNTAX

```
status report_qor
  [-significant_digits digits]
  [-scenarios scenario_list]
  [-summary]
  [-ignore_infeasible_paths]
```

Data Types

digits integer
scenario_list list

ARGUMENTS

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are reported. Allowed values are from 0 through 13. The default is 2 for all sections except the Area section, which has a default of 6. Specifying a valid value for significant digits overwrites the existing default. Using this option overrides the value set by the **report_default_significant_digits** variable.

-scenarios *scenario_list*

Reports the QoR for the specified scenarios of a design containing multiple scenarios. Inactive scenarios are skipped in the report. Each scenario is reported separately.

If you do not specify this option, the command reports the QoR on all active scenarios.

-summary

Reports a QoR summary without reporting details for each timing group. This option is supported only in DC Explorer and Design Compiler topographical mode.

-ignore_infeasible_paths

Ignores all paths flagged as infeasible during the latest compilation.

DESCRIPTION

This command reports timing-path group and cell count details, along with current design statistics such as combinational, noncombinational, and total area. The command also reports static power, design rule violations, and compile-time details.

Under the Cell Count section, the Leaf Cell Count report includes all leaf cells that are not constant cells. Constant cells are omitted in this count. The Combinational Cell Count and Sequential Cell Count only include leaf cells.

The design and scenario timing QoR that is reported counts an endpoint only one time when computing the total negative slack and the number of violation endpoints. This behavior is controlled by the **timing_report_union_tns** variable.

Multicorner-Multimode Support

By default, this command uses information from all active scenarios. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following example shows a QoR report:

```
prompt> report_qor
```

```
*****
Report : qor
Design : top
Version: E-2010.12
Date   : Tue Aug 3 15:12:52 2010
*****
```

Timing Path Group 'default'

```
-----
Levels of Logic:      23.00
Critical Path Length: 10.00
Critical Path Slack:  0.00
Critical Path Clk Period: Undef
Total Negative Slack: 0.00
No. of Violating Paths: 0.00
Worst Hold Violation: 0.00
Total Hold Violation: 0.00
No. of Hold Violations: 0.00
-----
```

Cell Count

```
-----
Hierarchical Cell Count: 15
Hierarchical Port Count: 708
Leaf Cell Count:        3072
Buf/Inv Cell Count:     411
CT Buf/Inv Cell Count:  0
Combinational Cell Count: 3072
Sequential Cell Count:  0
Macro Count:            0
-----
```

Area

```

-----
Combinational Area: 228281.600616
Noncombinational Area: 0.000000
Net Area:          0.000000
-----

```

```

Cell Area:      228281.600616
Design Area:    228281.600616

```

Design Rules

```

-----
Total Number of Nets:    3595
Nets With Violations:    0
-----

```

Hostname: host1

Compile CPU Statistics

```

-----
Resource Sharing:        9.62
Logic Optimization:     54.90
Mapping Optimization:    100.32
-----
Overall Compile Time:    176.09
Overall Compile Wall Clock Time: 178.56

```

1

The following is an example of a QoR report that uses the **-significant_digits** option to show 4 digits to the right of the decimal point:

prompt> **report_qor -significant_digits 4**

```

*****
Report : qor
Design : top
Version: E-2010.12
Date  : Fri Aug 6 17:34:51 2010
*****

```

Timing Path Group 'default'

```

-----
Levels of Logic:        26.0000
Critical Path Length:    9.9998
Critical Path Slack:     0.0002
Critical Path Clk Period:  Undef
Total Negative Slack:    0.0000
No. of Violating Paths:  0.0000
Worst Hold Violation:    0.0000
Total Hold Violation:    0.0000
No. of Hold Violations:  0.0000
-----

```

Cell Count

```

-----
Hierarchical Cell Count:    15
Hierarchical Port Count:    708
Leaf Cell Count:            3072
Buf/Inv Cell Count:         411

```

```

CT Buf/Inv Cell Count:      0
Combinational Cell Count:   3072
Sequential Cell Count:      0
Macro Count:                0
-----

```

Area

```

-----
Combinational Area:  228294.4006
Noncombinational Area:  0.0000
Net Area:            0.0000
-----

```

```

Cell Area:          228294.4006
Design Area:        228294.4006

```

Design Rules

```

-----
Total Number of Nets:    3591
Nets With Violations:    0
-----

```

Hostname: host1

Compile CPU Statistics

```

-----
Resource Sharing:        10.8724
Logic Optimization:      59.0820
Mapping Optimization:    110.3862
-----
Overall Compile Time:    192.1378
Overall Compile Wall Clock Time: 195.4446

```

1

The following is an example of a QoR report that uses the **-summary** option:

prompt> **report_qor -summary**

```

Scenario: wc_corner WNS: 1.40 TNS: 311.87 Number of Violating Paths: 2392
Scenario: bc_corner WNS: 0.00 TNS: 0.00 Number of Violating Paths: 0
Multi-Scenario WNS: 1.40 TNS: 311.87 Number of Violating Paths: 2392
Nets with DRC Violations: 352
Total moveable cell area: 315903.1
Total fixed cell area: 202475.1
Core area: (7800 3600 952200 691200)

Scenario: wc_corner (Hold) WNS: 0.47 TNS: 106.47 Number of Violating Paths: 466
Scenario: bc_corner (Hold) WNS: 0.79 TNS: 3963.11 Number of Violating Paths: 18290
Multi-Scenario (Hold) WNS: 0.79 TNS: 4069.59 Number of Violating Paths: 18756

```

1

SEE ALSO

```
timing_report_union_tns(3)  
report_default_significant_digits(3)
```

report_qtm_model

Reports Quick Timing Model (QTM) data.

SYNTAX

```
string report_qtm_model  
  [-global_parameters]  
  [-ports]  
  [-arcs]
```

ARGUMENTS

-global_parameters

Specifies that global parameters of the current QTM design be reported.

-ports

Specifies that ports of the current QTM design be reported.

-arcs

Specifies that timing arcs of the current QTM design be reported.

DESCRIPTION

The **report_qtm_model** command prints a report of the active QTM model. The information details the global parameters, ports, and arcs of the QTM model. By default, the command prints out the report of global parameters, ports, and arcs of the model.

If you want to see the global parameters in the QTM model, use the **-global_parameters** option.

If you want to see only the ports in the QTM model, use the **-ports** option.

If you want to see the arcs in the QTM model, use the **-arcs** option.

For a basic description of QTM, see the **create_qtm_model** manual page. For a more detailed description of QTM, see the *ICC Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command reports all the attributes of the currently active QTM model.

```
prompt> report_qtm_model
```

```
*****
Report : qtm_model
Design : test_lib
*****

create_qtm_model test_lib
set_qtm_technology -library slow -max_transition 1.0
set_qtm_global_parameter -param clk_to_output -value 1.0
set_qtm_global_parameter -param setup -value 0.5
set_qtm_global_parameter -param hold -value 0.0
create_qtm_path_type -lib_cell AN210 -input_pin A -output_pin Y -fanout 2 path1
create_qtm_drive_type -lib_cell AN210 -input_pin A -output_pin Y drive1
create_qtm_load_type -lib_cell AN210 -input_pin A load1
create_qtm_port -type clock { CLK }
create_qtm_port -type input { A }
create_qtm_port -type input { B }
create_qtm_port -type input { X }
create_qtm_port -type input { Y }
set_qtm_port_load -type load1 -factor 2 { A }
set_qtm_port_load -value 1.0 { B }
set_qtm_port_drive -type drive1 -type drive1 { X }
set_qtm_port_drive -value 1.0 { Y }
create_qtm_constraint_arc -setup -from CLK -edge rise -value 1.0 -to { A }
create_qtm_constraint_arc -setup -from CLK -edge rise -path_type path1 -path_factor 2 -to { A }
create_qtm_delay_arc -from { CLK } -to { X } -edge rise -value 1.0
create_qtm_delay_arc -from { CLK } -to { Y } -edge rise -path_type path1 -path_factor 7
report_qtm_model
```

The following command reports the global parameters of the currently active QTM model.

```
prompt> report_qtm_model -global_parameters
```

```
*****
Report : qtm_model
Design : test_lib
*****

create_qtm_model test_lib
set_qtm_technology -library slow -max_transition 1.0
set_qtm_global_parameter -param clk_to_output -value 1.0
set_qtm_global_parameter -param setup -value 0.5
set_qtm_global_parameter -param hold -value 0.0
create_qtm_path_type -lib_cell AN210 -input_pin A -output_pin Y -fanout 2 path1
create_qtm_drive_type -lib_cell AN210 -input_pin A -output_pin Y drive1
create_qtm_load_type -lib_cell AN210 -input_pin A load1
report_qtm_model
```

SEE ALSO

`create_qtm_model(2)`

`save_qtm_model(2)`

report_reference

Displays information about references in the current instance or in the current design.

SYNTAX

```
status report_reference
      [-nosplit]
      [-hierarchy]
```

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-hierarchy

Displays information about references across the hierarchy in the current instance or the current design.

DESCRIPTION

This command displays information about all references in the current instance or the current design. If the current instance has been set, the report is generated for the design of that instance. Otherwise the report is generated for the current design.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following is an example of a reference report:

```
prompt> report_reference
```

```
*****
```

```
Report : reference
```

Design : CONTROL
 Version: Y-2006.06
 Date : Mon Dec 12 14:09:33 2005

Attributes:

b - black box (unknown)
 bo - boundary optimization
 d - dont_touch
 h - hierarchical
 n - noncombinational
 r - removable
 s - synthetic module
 u - contains unmapped logic

Reference	Library	Unit Area	Count	Total Area	Attributes
CTLX		101.00	1	101.00	h
INV	tech_lib	1.00	1	1.00	
JMM		0.00	0	0.00	b, d
JMM1		0.00	0	0.00	b
JPMP		0.00	1	0.00	h, u
MX1P	tech_lib	8.00	8	64.00	n
NAND2	tech_lib	1.00	6	6.00	
NAND3	tech_lib	2.00	1	2.00	
Total 8 references				174.00	

The following is an example of a reference report across a hierarchy:

prompt> **report_reference -hierarchy**

 Report : reference
 Design : top
 Version: Y-2006.06
 Date : Mon Dec 12 04:25:17 2005

Attributes:

b - black box (unknown)
 bo - boundary optimization
 d - dont_touch
 h - hierarchical
 n - noncombinational
 r - removable
 s - synthetic module
 u - contains unmapped logic

Reference	Library	Unit Area	Count	Total Area	Attributes
AN3	tech_lib	2.00	8	16.00	
mid_0		56.00	1	56.00	h, n
Total 2 references				72.00	

Design: mid_0

Reference	Library	Unit Area	Count	Total Area	Attributes
low_0		28.00	1	28.00	h, n
low_5		28.00	1	28.00	h, n
Total 2 references			56.00		

Design: low_0

Reference	Library	Unit Area	Count	Total Area	Attributes
FD1	tech_lib	7.00	4	28.00	n
Total 1 references			28.00		

Design: low_5

Reference	Library	Unit Area	Count	Total Area	Attributes
FD1	tech_lib	7.00	4	28.00	n
Total 1 references			28.00		

SEE ALSO

report_cell(2)
report_design(2)

report_resources

Lists the resources and datapath blocks used in the design of the current instance, in the current design, or in a specific design or list of designs.

SYNTAX

```
status report_resources
  [-nosplit]
  [-hierarchy]
  [-context]
  [-minpower]
  [-html_file_name filename]
  [design_list]
```

Data Types

```
filename    string
design_list  string
```

ARGUMENTS

-nosplit

Prevents line splitting. Most design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-hierarchy

Reports information about all resources used in the hierarchy of the current design or the current instance. By default, the tool reports only the resources used in the top level of the current design or the current instance. The **-hierarchy** option is ignored if you specify a list of designs with the *design_list* argument.

-context

Reports context information in the DesignWare generator. Currently, the power context information is printed out.

-minpower

Reports only DesignWare cells that are optimized for the minpower flow.

-html_file_name *filename*

Redirects the report to a file in HTML format.

design_list

Reports resources for a specific design or list of designs. You cannot specify the *design_list* argument and the **-hierarchy** option together. In this case, the **-hierarchy** option is ignored, Design Compiler issues a warning message, and it reports only the first level of hierarchy for the specified design or designs.

DESCRIPTION

This command lists the resources and datapath blocks used in the design of the current instance, in the current design, or in a specific design or list of designs. If the current instance is set, the report is generated for the design of that instance. Otherwise, the report is generated for the current design. To specify a specific design or a list of designs, use the *design_list* argument.

A resource is an arithmetic or comparison operator read in as part of an HDL design. Resources can be shared when compiling the design and are reported after the compile operation completes.

A datapath block contains one or more resources that are grouped together and optimized by a datapath generator.

This report also has information on the parameters used to build the module, user-declared resources in each module, shared operations, and resources that are transformed into datapath blocks.

A detailed report is printed in the DC Ultra flow. The report displays the following information about the datapath blocks in the design:

- Design name
- Source file location
- Input and output connections for each datapath operand and bit width
- Operation name with source line number
- Datapath operation or expression

EXAMPLES

The following example displays resource information for the current design:

```
prompt> report_resources
```

```
*****
Report : resources
Design : REGCNT
Version: A-2007.12-SP1
Date  : Fri Jan  4 01:59:13 2008
*****
```

```
Resource Sharing Report for design REGCNT in file
/am/remote/cae7/testcases/DAC94/suite/hadv/src2/verilog/regcnt.v
```

```
=====
| Cell      | Module    | Parameters | Contained Operations |
=====
| sub_x_0   | DW01_dec  | width=5   | sub_6                 |
=====
```


Implementation Report

		Current	Set	
Cell	Module	Implementation	Implementation	
sub_x_0	DW01_dec	rpl		

The following example shows the datapath extraction report for the design named *mult_add*. As shown in the example, the Datapath Report has two tables for each datapath block. The first table shows the datapath block along with the contained operations. The second table shows datapath expressions for the block along with the type (Primary Input, Primary Output, or Internal FanOut), signage, and bit width of each variable in the expression.

prompt> **report_resources**

```
*****
Report : resources
Design : mult_add
Version: A-2007.12-SP1
Date  : Fri Jan  4 01:59:13 2008
*****
```

Resource Report for this hierarchy in file
/remote/disks/user/dp_report/dp_test.v

Cell	Module	Parameters	Contained Operations	
DP_OP_4_296	DP_OP_4_296			

Datapath Report for DP_OP_4_296

Cell	Contained Operations	
DP_OP_4_296	mult_16 add_18 mult_17	

Var	Type	Class	Width	Expression	
I1	PI	Unsigned	4		
I2	PI	Unsigned	4		
I3	PI	Unsigned	4		
I4	PI	Unsigned	4		
T0	IFO	Unsigned	8	I1 * I2	
T1	IFO	Unsigned	8	I3 * I4	
O1	PO	Unsigned	8	T0 + T1	

Implementation Report

		Current	Set	
Cell	Module	Implementation	Implementation	

```
| DP_OP_4_296 | DP_OP_4_296 | str | |
=====
```

The following example displays resource and datapath information for the current design:

```
prompt> report_resources -hierarchy
```

```
*****
Report : resources
Design : trunc3
Version: A-2007.12-SP1
Date  : Fri Jan 4 02:33:50 2008
*****
```

Resource Report for this hierarchy in file ./designs/trunc3.v

```
=====
| Cell      | Module      | Parameters | Contained Operations |
=====
| add_x_9_0 | DW01_add    | width=6    | add_9                 |
| DP_OP_6_296 | DP_OP_6_296 |           |                       |
=====
```

Datapath Report for DP_OP_6_296

```
=====
| Cell      | Contained Operations |
=====
| DP_OP_6_296 | sub_8 add_8          |
=====
```

```
=====
| | | Data | |
| Var | Type | Class | Width | Expression |
=====
| I1 | PI | Unsigned | 5 | |
| I2 | PI | Unsigned | 5 | |
| I3 | PI | Unsigned | 5 | |
| O1 | PO | Signed | 7 | I1 - I2 + I3 |
=====
```

Implementation Report

```
=====
| | | Current | Set |
| Cell | Module | Implementation | Implementation |
=====
| DP_OP_6_296 | DP_OP_6_296 | str | |
| add_x_9_0 | DW01_add | rpl | |
=====
```

No multiplexors to report

```
*****
Design : trunc3_DW01_add_0
*****
```

No resource sharing information to report.

No implementations to report

No multiplexors to report

```
*****
Design : trunc3_DP_OP_6_296_0
*****
```

No resource sharing information to report.

No implementations to report

No multiplexors to report

The following example displays resource, context, and smartgen information for the minpower flow. The Datapath Optimization Report shows the smartgen options set by the user.

prompt> **report_resources -hierarchy**

```
*****
Design : firpolyint_IN_REG1
*****
```

Minpower implementation Report

```
=====
|          |          | Current   | Set      |
| Cell      | Module    | Implementation | Implementation |
=====
| DP_OP_24_296_7745 | DP_OP_24_296_7745 | str (power)   |          |
=====
```

Generator Context Report for DP_OP_24J1_296_7745

```
=====
| Input |      |          | With |
| Ports | Width | Variable Inputs | Switching Activity |
=====
| I1 | 12 | 12 100% | 12(12*) 100% |
=====
```

Note: * indicates the number of bits that use default switching activity

Datapath Optimization Report

```
=====
|          | Optimization |          |
| Cell      | Mode         | Smartgen Options      |
=====
| sub_x_870_1 | area        | powerEffort:medium    |
| lt_x_870_3  | area        | powerEffort:medium    |
=====
```

SEE ALSO

compile(2)
compile_ultra(2)

report_retention_cell

Displays information about retention cells in the current scope.

SYNTAX

```
status report_retention_cell
  [retention_cells]
  [-domain power_domains]
  [-retention_strategy retention_strategy_names]
  [-clamps]
  [-verbose]
```

Data Types

```
retention_cells      list
power_domains        list
retention_strategy_names string
```

ARGUMENTS

retention_cells

Specifies the retention cells that are to be reported. If the specified cells do not exist in the current scope, the command fails. By default, all retention cells in the current scope are reported when no other options are specified.

-domain *power_domains*

Specifies the power domains to report all retention cells in the design belonging to the power domains. If the specified power domains do not exist in the current scope, the command fails.

-retention_strategy *retention_strategy_names*

Specifies the names of the retention strategies to report all retention cells in the design that belong to the given retention strategies. When specifying the retention strategy, you must also specify a single power domain with the **-domain** option. If the specified retention strategy is not a part of the specified power domain, the command fails.

-clamps

Report clamp cells associated to zero-pin retention cells

-verbose

Reports retention strategy details in addition to information about the retention cells.

DESCRIPTION

The **report_retention_cell** command provides detailed information of all retention cells in the current scope. If the **-domain** option is specified, the tool reports on all retention cells in the specified domains. If the **-retention_strategy** option is specified, the tool reports on all retention cells under the specified strategy for the given power domain.

The report shows the instance name of the retention cell, the library reference name of the retention cell, and the retention strategy. If the **-verbose** option is specified, the report also shows the details of the retention strategy, which include the name of the retention strategy, the names of the save and restore signals, the save sense and restore sense values, and the names of the power and ground nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the information about all of the retention cells in the current scope:

```
prompt> report_retention_cell
```

```
-----
Power Domain : PD1

=====
| Ret. Cell      | Ret. Lib. Cell  | Strategy      |
=====
q2_reg          | RTCLDFFPQ_F1_SNPM | retention_1    |
q1_reg          | RTCLDFFPQ_F1_SNPM | retention_1    |
=====

-----
1
```

The following example reports detailed information about all of the retention cells in the current scope:

```
prompt> report_retention_cell -verbose
```

```
-----
Power Domain : PD1

=====
| Ret. Cell      | Ret. Lib. Cell  | Strategy      |
=====
regout_reg[0]    | RTCLDFFPQ_F1_SNPM | retention_strategy_1 |
regout_reg[1]    | RTCLDFFPQ_F1_SNPM | retention_strategy_1 |
regout_reg[2]    | RTCLDFFPQ_F1_SNPM | retention_strategy_1 |
regout_reg[3]    | RTCLDFFPQ_F1_SNPM | retention_strategy_1 |
=====

=====
```

```
| Strategy      | Save | Save | Restore | Restore | Power | Ground |
|              | Signal | Sense | Signal | Sense | Net | Net |
=====
retention_strategy_1 | SAVE1 | High | RESTORE1 | High | SN1 | - |
=====
```

1

SEE ALSO

- map_retention_cell(2)
- report_isolation_cell(2)
- report_level_shifter(2)
- set_retention(2)
- set_retention_control(2)

report_retention_clamp_cell

Displays information about retention cells in the current scope.

SYNTAX

```
status report_retention_clamp_cell
  [retention_cells]
  [-domain power_domains]
  [-retention_strategy retention_strategy_names]
  [-verbose]
```

Data Types

<i>retention_cells</i>	list
<i>power_domains</i>	list
<i>retention_strategy_names</i>	string

ARGUMENTS

retention_cells

Specifies the retention cells that are to be reported. If the specified cells do not exist in the current scope, the command fails. By default, all retention cells in the current scope are reported when no other options are specified.

-domain *power_domains*

Specifies the power domains to report all retention cells in the design belonging to the power domains. If the specified power domains do not exist in the current scope, the command fails.

-retention_strategy *retention_strategy_names*

Specifies the names of the retention strategies to report all retention cells in the design that belong to the given retention strategies. When specifying the retention strategy, you must also specify a single power domain with the **-domain** option. If the specified retention strategy is not a part of the specified power domain, the command fails.

-verbose

Reports retention strategy details in addition to information about the retention cells.

DESCRIPTION

The **report_retention_cell** command provides detailed information of all retention cells in the current scope. If the **-domain** option is

specified, the tool reports on all retention cells in the specified domains. If the **-retention_strategy** option is specified, the tool reports on all retention cells under the specified strategy for the given power domain.

The report shows the instance name of the retention cell, the library reference name of the retention cell, and the retention strategy. If the **-verbose** option is specified, the report also shows the details of the retention strategy, which include the name of the retention strategy, the names of the save and restore signals, the save sense and restore sense values, and the names of the power and ground nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the information about all of the retention cells in the current scope:

```
prompt> report_retention_cell
```

```
-----
Power Domain : PD1

=====
| Ret. Cell      | Ret. Lib. Cell  | Strategy      |
=====
| q2_reg         | RTCLDFFPQ_F1_SNPM | retention_1    |
| q1_reg         | RTCLDFFPQ_F1_SNPM | retention_1    |
=====

-----
1
```

The following example reports detailed information about all of the retention cells in the current scope:

```
prompt> report_retention_cell -verbose
```

```
-----
Power Domain : PD1

=====
| Ret. Cell      | Ret. Lib. Cell  | Strategy      |
=====
| regout_reg[0]  | RTCLDFFPQ_F1_SNPM | retention_strategy_1 |
| regout_reg[1]  | RTCLDFFPQ_F1_SNPM | retention_strategy_1 |
| regout_reg[2]  | RTCLDFFPQ_F1_SNPM | retention_strategy_1 |
| regout_reg[3]  | RTCLDFFPQ_F1_SNPM | retention_strategy_1 |
=====

=====
| Strategy      | Save | Save | Restore | Restore | Power | Ground |
|               | Signal | Sense | Signal | Sense | Net | Net |
=====
| retention_strategy_1 | SAVE1 | High | RESTORE1 | High | SN1 | - |
```

=====

1

SEE ALSO

map_retention_cell(2)
report_isolation_cell(2)
report_level_shifter(2)
set_retention(2)
set_retention_control(2)

report_route_zrt_common_options

Reports the settings of route options that are common among the Zroute router commands.

SYNTAX

status **report_route_zrt_common_options**

ARGUMENTS

The **report_route_zrt_common_options** command has no arguments.

DESCRIPTION

This command reports the settings of router options that have been set by the **set_route_zrt_common_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command reports the settings of the common route options.

```
prompt> report_route_zrt_common_options
```

SEE ALSO

set_route_zrt_common_options(2)
get_route_zrt_common_options(2)

report_route_zrt_global_options

Reports the settings of the global router options.

SYNTAX

status **report_route_zrt_global_options**

ARGUMENTS

The **report_route_zrt_global_options** command has no arguments.

DESCRIPTION

This command reports the settings of the global router options that have been set by the **set_route_zrt_global_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the global router settings.

```
prompt> report_route_zrt_global_options
```

SEE ALSO

get_route_zrt_global_options(2)
set_route_zrt_global_options(2)

report_routing_rules

Reports on design-specific nondefault routing rules defined by the **define_routing_rule** command.

SYNTAX

```
status report_routing_rules  
    [rule_name]  
    [-output file_name]
```

Data Types

```
rule_name    string  
file_name    string
```

ARGUMENTS

rule_name

Specifies the name of the design-specific nondefault routing rule to report on. By default, the tool reports all design-specific nondefault routing rules.

-output *file_name*

Generates a Tcl script that contains the **define_routing_rule** commands that define the specified nondefault routing rules.

DESCRIPTION

This command reports on design-specific nondefault routing rules defined by the **define_routing_rule** command. It does not report rules defined in the physical library.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports on the design-specific nondefault rule named WideMetal.

```
prompt> report_routing_rules WideMetal
```

SEE ALSO

`define_routing_rule(2)`
`set_net_routing_rule(2)`

report_rp_group_options

Reports relative placement group attributes on the specified relative placement groups.

SYNTAX

```
status report_rp_group_options  
rp_groups
```

Data Types

rp_groups collection or list

ARGUMENTS

rp_groups

Specifies the relative placement groups for which you want to report the attributes. If relative placement groups are not specified, the attributes for all relative placement groups will be reported.

DESCRIPTION

The **report_rp_group_options** command reports the attributes of the specified relative placement groups. The attributes were set by using either the **create_rp_group** or the **set_rp_group_options** command.

This command is supported only for designs that do not contain multiply-instantiated designs.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **report_rp_group_options** to report the attributes for a relative placement group:

```
prompt> report_rp_group_options {example3::top_group example3::block2}
```

```
*****
```

```
Report : The RP group options.
```

```
Design : example3
Version: Y-2006.06
Date   : Fri Jun  9 00:58:54 2006
*****
RP group: example3::top_group
utilization   : 1.000000
ignore        : FALSE
x_offset      : 0
y_offset      : 0
cts_option    : fixed_placement
RP group: example3::top_group
utilization   : 1.000000
ignore        : FALSE
1
```

SEE ALSO

```
create_rp_group(2)
set_rp_group_options(2)
```

report_saif

Reports the statistics of switching activity annotation, on the current design or instance.

SYNTAX

```
status report_saif
  [-only cell_or_net_list]
  [-hierarchy]
  [-missing]
  [-annotated_flag]
  [-rtl_saif]
```

Data Types

cell_or_net_list list

ARGUMENTS

-only *cell_or_net_list*

Specifies that switching activity annotation is to be reported only for the specified object list.

-hierarchy

Specifies that the command reports the switching activity on the entire design hierarchy, starting with the current instance. If the **-hierarchy** option is not used, switching activity is reported only on the design objects in the current hierarchy of the current instance.

-missing

Specifies that the report should list the design elements that do not have user specified switching activity annotation. This report does not include constant value nets (logic 1 or logic 0 nets), and pins and ports connected to such nets. These types of nets, pins, and ports are annotated with the default switching activity if they are not user-annotated.

-annotated_flag

Specifies that design elements that have user-annotated switching activity should be reported.

-rtl_saif

Specifies that switching activity annotations should be reported for objects annotated by an RTL backward SAIF file.

DESCRIPTION

The **report_saif** command reports the switching activity annotation on the nets, ports, and cells in the current design or instance.

The **report_saif** command can be used to display switching activity annotation of synthesis invariant objects by using the **-rtl_saif** flag. The percentage of user annotation of sequential cells could decrease after compile due to the registers generated from DesignWare. If this option is not used, then **report_saif** command displays information on the design ports, nets, and pins of non-hierarchical cells.

The **report_saif** command reports the percentage of objects with

switching activity that is user-annotated, default-annotated, and propagated. Switching activity can be annotated by the user by using the **set_switching_activity**, **read_saif**, and **merge_saif** commands.

During power calculations, the tool estimates the switching activity of objects that are not user-annotated by propagating the user or default annotated switching activity. Switching activity is annotated with default values on objects whose switching activity can be derived accurately, such as clocks and nets driven by constants, or on objects that cannot be annotated using the propagation mechanism, such as the primary inputs of the design and outputs of black-box cells. Switching activity is annotated by default and propagated automatically by the **report_power** command.

Propagated switching activity is less accurate than switching activity obtained by simulation. So the percentage values in the user-annotated column of the report generated by **report_saif** are an indication of the accuracy of the power reports.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following examples report switching activity for the current design.

In the following example, the **report_saif** command is used after the **read_saif** command, on a gate-level design. Because the **-missing** option is used, the report shows the nonannotated ports, nets, and pins.

```
prompt> report_saif -hierarchy -missing
```

```
*****
```

```
Report : saif
```

```
-hierarchy
```

```
-missing
```

```
Design : addr_con
```

```
Version: ...
```

```
Date : ...
```

```
*****
```

```
-----
      User      Default      Propagated
Object type Annotated (%) Annotated (%) Activity (%) Total
-----
Nets      251(99.21%)  1(0.40%)   1(0.40%)   253
```

```

Ports    59(98.33%)  1(1.67%)  0(0.00%)  60
Pins     251(99.60%)  0(0.00%)  1(0.40%)  252
-----

```

List of nonannotated ports :

```
mem_config(0)
```

List of nonannotated nets :

```
n677
net41
```

List of nonannotated pins :

```
U519/A
```

In the following example, pin and net annotations are reported only for the cell `{last_addr_regx6x}`. One of the pins of the cell and the corresponding net are not annotated.

```
prompt> report_saif -missing -only {last_addr_regx6x}
```

```

*****
Report : saif
        -missing
        -only
Design : addr_con
Version: ...
Date   : ...
*****

```

```

-----
      User      Default      Propagated
Object type Annotated (%) Annotated (%) Activity (%) Total
-----
Nets        3(75.00%)   0(0.00%)   0(0.00%)    4
Ports       0(0.00%)   0(0.00%)   0(0.00%)    0
Pins        3(75.00%)   0(0.00%)   0(0.00%)    4
-----

```

List of nonannotated nets :

```
n677
```

List of nonannotated pins :

```
last_addr_regx6x/Q
```

The following example generates a switching activity report on synthesis invariant objects:

```
prompt> report_saif -rtl_saif
```

```

*****
Report : saif
        -rtl_saif
Design : cpu
Version: ...

```

Date : ...

	User	Default	Propagated	
Object type	Annotated (%)	Annotated (%)	Activity (%)	Total

Ports	13(100.00%)	0(0.00%)	0(0.00%)	13
Seq Cells	620(99.68%)	0(0.00%)	0(0.00%)	622
Tri Cells	36(90.00%)	0(0.00%)	0(0.00%)	40

SEE ALSO

- read_saif(2)
- merge_saif(2)
- report_power(2)
- set_switching_activity(2)

report_scaling_lib_group

Generates a report of scaling library groups previously created by the **define_scaling_lib_group** command.

SYNTAX

```
status report_scaling_lib_group
  [-all]
  [-nosplit]
  [-show_list show_list]
  [-object_list object_list]
```

Data Types

```
show_list    list
object_list  list
```

ARGUMENTS

-all

Reports a list of all scaling library groups defined with the **define_scaling_lib_group** command, including any that are not currently being used. By default, the command reports only the scaling library groups being used in the current design.

-nosplit

Specifies not to split lines when column fields overflow.

-show_list *show_list*

Specifies the types of detailed library information to include in the report. The *show_list* can contain any one or more of the following items: **process**, **voltage**, **temperature**, and **extended_name**. The **extended_name** keyword reports the full directory path to the library name. By default, these types of information are not reported.

-object_list *object_list*

Specifies the cells for which to report the scaling library groups. Only the scaling groups that have been set on these objects are reported. By default, the command reports all scaling groups used in the current design.

DESCRIPTION

The **report_scaling_lib_group** command reports the scaling library groups defined by the **define_scaling_lib_group** command and set for usage with the **set_scaling_lib_group** command.

By default, the **report_scaling_lib_group** command reports the scaling groups for the current scenario of the current design. If you use the **-object_list** option, it reports the scaling groups for the specified objects in the current scenario.

The **-all** option reports all the scaling library groups defined by the **define_scaling_lib_group** command, either for the current design or a list of specified objects.

You can view detailed information about the libraries by using the **-show_list** option.

Multicorner-Multimode Support

This command applies to the current scenario only by default; when **-all** option is specified, this command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the scaling relationships that exist in the design.

```
prompt> report_scaling_lib_group
...
Group  Library
-----
GRP_7
    RC03BTH3_max
    RC03BTH3_max
GRP_8
    RC03BTH3_max
    RC03BTH3_max
-----
1
```

The following example uses the **-show_list** option to show more detailed library information.

```
prompt> report_scaling_lib_group -show_list {voltage temperature process}
...
Group  Library    Temperature  Voltage    Process
-----
GRP_10
    RC03BTH3_max -40.000000  0.800000  1.000000
    RC03BTH3_max 125.000000  1.050000  1.000000
-----
1
```

SEE ALSO

- define_scaling_lib_group(2)
- remove_scaling_lib_group(2)
- report_lib(2)
- set_scaling_lib_group(2)

report_scan_chain

Reports the SCANDEF scan chains defined on the current design.

SYNTAX

status **report_scan_chain**

ARGUMENTS

The **report_scan_chain** command has no arguments.

DESCRIPTION

This command reports all scan chain information. The chains are defined on the current open CEL.

Using SCANDEF data, the report displays each chain's start and stop ports or pins, partition label, validation status, and all scan chain components. The scan chain components are shown in the order in which they appear in the chain of the netlist, from the start point to the stop point.

The status can be NOT YET VALIDATED, VALIDATED/V or FAILED/F.

- NOT YET VALIDATED indicates that the **check_scan_chain** command has not been run to determine the status.
- VALIDATED indicates that the SCANDEF data in the report is consistent with the netlist.
- FAILED indicates an inconsistency between the SCANDEF data and the netlist.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of a scan chain report:

```
prompt> report_scan_chain
```

Report : Scan Chain
Design : CORE
Version: W-2004.12
Date : Thu Jun 2 14:02:00 2005

SCANCHAIN : ScanGroup_zw_test_test_si1
START : test_si1
STOP : U1/LOCKUP/D
PARTITION : CLK1_45_45
STATUS : VALIDATED

U1/scrs_reg (IN D1) (OUT Q)
U1/dcrs_reg (IN D1) (OUT Q)

SCANCHAIN : ScanGroup_U1/LOCKUP_Q
START : U1/LOCKUP/Q
STOP : U1/LOCKUP1/D
PARTITION : CLK1_45_45
STATUS : VALIDATED

U1/rrst_L_reg (IN D1) (OUT Q)
U1/stxen_reg (IN D1) (OUT Q)

1

SEE ALSO

report_scan_compression_configuration

Displays options specified by the **set_scan_compression_configuration** command.

SYNTAX

status **report_scan_compression_configuration**
[-test_mode *mode_name_list*]

Data Types

mode_name_list string

ARGUMENTS

-test_mode *mode_name_list*

Specifies the test mode(s) to report. If not specified, the tool reports on the current test mode, as displayed by the **current_test_mode** command. If no test modes were created, the tool reports on the default compression mode. If **all** is specified, the tool displays the configurations of all test modes.

DESCRIPTION

This command displays options specified by the **set_scan_compression_configuration** command on the current design.

The command reports the following information:

- Minimum compression
- Type of integration (if any)
- Level of X-tolerance
- Static X-chain creation
- Compression chain synchronization

- Diagnosis accuracy
- Number of decompressor inputs
- Number of compressor outputs
- Compression mode chain count
- Maximum chain length

EXAMPLES

The following is an example of a scan compression configuration report:

```
prompt> report_scan_compression_configuration
```

```
*****
```

```
Report : Scan Compression Configuration
```

```
Design : des_unit
```

```
Version: I-2013.12
```

```
Date   : Mon Oct 21 09:19:47 2013
```

```
*****
```

```
Minimum Compression    50
Integration             False
hybrid                 False
xtolerance              default
static_x_chain_isolation False
synchronize_chains     none
force_diagnosis        False
Inputs                 Unspecified
Outputs                Unspecified
Chain Count            Unspecified
Maximum Chain Length   Unspecified
```

```
1
```

SEE ALSO

```
current_test_mode(2)
reset_scan_compression_configuration(2)
set_scan_compression_configuration(2)
```

report_scan_configuration

Displays options specified by the **set_scan_configuration** command.

SYNTAX

```
status report_scan_configuration  
[-test_mode mode_name_list]
```

Data Types

mode_name_list list

ARGUMENTS

-test_mode *mode_name_list*

Specifies the test mode(s) to report. If not specified, the tool reports the scan configuration of the current test mode, as displayed by the **current_test_mode** command. If no test modes were created, the tool reports the scan configuration for the default test mode. If **all** is specified, the tool displays scan configuration information for all test modes.

DESCRIPTION

This command displays the options specified by the **set_scan_configuration** command on the current design. For more information on what is reported, see the EXAMPLES section.

EXAMPLES

The following is an example of a scan configuration report for the default test mode in a flow where no user-defined test modes are defined:

```
prompt> report_scan_configuration
```

```
*****
```

```
Report : Scan configuration
```

```
Design : top
```

```
Version: I-2013.12-SP1
```

```
Date   : Fri Jan 3 11:28:25 2014
```

=====

TEST MODE: all_dft

VIEW : Specification

=====

Chain count: 5
 Scan Style: Multiplexed flip-flop
 Maximum scan chain length: Undefined
 Exact scan chain length: Undefined
 Physical Partitioning: Horizontal
 Replace: True
 Preserve multibit segments: False
 Clock mixing: No mix
 Internal clocks: none
 Add lockup: True
 Lockup type: latch
 Insert terminal lockup: False
 Create dedicated scan out ports: False
 Shared scan in: 0
 Bidirectional mode: No bidirectional type
 Internal Clock Mixing: False
 Test Clocks by System Clocks: False
 Hierarchical Isolation: False
 Multiple Scan Enable: Disable
 Pipeline Scan Enable: Disable
 Voltage Mixing: False
 Identify Shift Register: False
 Power Domain Mixing: False
 Reuse MV Isolation Cells: True
 Multi LSSD: Disable

The following is an example of a scan configuration report for all user-defined test modes:

prompt> **report_scan_configuration -test_mode all**

Report : Scan configuration

Design : sub

Version: I-2013.12-SP1-CS2

Date : Fri Jan 3 11:35:14 2014

=====

TEST MODE: all_dft

VIEW : Specification

=====

Chain count: Undefined
 Scan Style: Multiplexed flip-flop
 Maximum scan chain length: Undefined
 Exact scan chain length: Undefined
 Physical Partitioning: Horizontal
 Replace: True
 Preserve multibit segments: False
 Clock mixing: No mix
 Internal clocks: none
 Add lockup: True

```

Lockup type:          latch
Insert terminal lockup: False
Create dedicated scan out ports: False
Shared scan in:       0
Bidirectional mode:   No bidirectional type
Internal Clock Mixing: False
Test Clocks by System Clocks: False
Hierarchical Isolation: False
Multiple Scan Enable: Disable
Pipeline Scan Enable: Disable
Voltage Mixing:       False
Identify Shift Register: False
Power Domain Mixing:  False
Reuse MV Isolation Cells: True
Multi LSSD:           Disable

```

```

=====
TEST MODE: MY_MODE1

```

```

VIEW : Specification

```

```

=====
Chain count:          50
Scan Style:           Multiplexed flip-flop
Maximum scan chain length: Undefined
Exact scan chain length: Undefined
Physical Partitioning: Horizontal
Replace:              True
Preserve multibit segments: False
Clock mixing:         Mix clocks
Internal clocks:      none
Add lockup:           True
Lockup type:          latch
Insert terminal lockup: False
Create dedicated scan out ports: False
Shared scan in:       0
Bidirectional mode:   No bidirectional type
Internal Clock Mixing: False
Test Clocks by System Clocks: False
Hierarchical Isolation: False
Multiple Scan Enable: Disable
Pipeline Scan Enable: Disable
Voltage Mixing:       False
Identify Shift Register: False
Power Domain Mixing:  False
Reuse MV Isolation Cells: True

```

```

=====
TEST MODE: MY_MODE2

```

```

VIEW : Specification

```

```

=====
Chain count:          200
Scan Style:           Multiplexed flip-flop
Maximum scan chain length: Undefined
Exact scan chain length: Undefined
Physical Partitioning: Horizontal
Replace:              True
Preserve multibit segments: False

```

Clock mixing:	No mix
Internal clocks:	none
Add lockup:	True
Lockup type:	latch
Insert terminal lockup:	False
Create dedicated scan out ports:	False
Shared scan in:	0
Bidirectional mode:	No bidirectional type
Internal Clock Mixing:	False
Test Clocks by System Clocks:	False
Hierarchical Isolation:	False
Multiple Scan Enable:	Disable
Pipeline Scan Enable:	Disable
Voltage Mixing:	False
Identify Shift Register:	False
Power Domain Mixing:	False
Reuse MV Isolation Cells:	True

SEE ALSO

current_test_mode(2)
report_dft_configuration(2)
report_dft_signal(2)
report_scan_path(2)
report_scan_configuration(2)
set_dft_configuration(2)
set_dft_signal(2)
set_scan_configuration(2)
set_scan_path(2)

report_scan_group

Reports all scan groups that were specified using the **set_scan_group** command.

SYNTAX

```
status report_scan_group  
      scan_group_names
```

Data Types

```
scan_group_names  string
```

ARGUMENTS

scan_group_names

Specifies the names of scan groups that you want to report. The scan groups must be previously defined using the **set_scan_group** command.

DESCRIPTION

This command reports all scan groups that were specified using the **set_scan_group** command. If no names are specified, all previously-defined scan groups are reported.

If you specify the name of a scan group to report, **report_scan_group** also reports the names of the elements within that group.

If the name you specify has not been previously defined as a scan group, the command issues a warning and exits.

EXAMPLES

The following example show how to use the **report_scan_group** command:

```
prompt> report_scan_group group1
```

```
prompt> report_scan_group all
```

```
prompt> report_scan_group
```

SEE ALSO

insert_dft(2)
preview_dft(2)
remove_scan_group(2)
set_scan_group(2)

report_scan_link

Reports the scan links specified with the **set_scan_link** command.

SYNTAX

```
status report_scan_link  
-test_mode mode_name  
[link_type]
```

Data Types

mode_name string

ARGUMENTS

-test_mode *mode_name*

Specifies the name of the mode for which the specifications are reported. The default is current mode.

DESCRIPTION

The **report_scan_link** command reports scan links specified with the **set_scan_link** command. Use the **-test_mode** option to report the scan links specified in a given mode. If the **-test_mode** option is not specified, the scan links in the current mode are reported.

Scan links connect scan cells, scan segments, and scan ports within scan chains. The tool supports scan links that are implemented as wires and scan-out lock-up latches.

Use the **report_scan_link** command to view the specified scan links.

To review your scan link specifications, use the **preview_dft** command with the **-show** option set to **cells**. To create scan links and add them to the current design, use the **insert_dft** command.

EXAMPLES

The following example declares a scan-out lock-up latch named *a_lckup* and then removes it with the **remove_scan_link** command:

```
prompt> current_design WC66
```

```
prompt> set_scan_link a_lckup Lockup
```

```
prompt> report_scan_link
```

```
=====
TEST MODE: Internal_scan
VIEW   : Specification
=====
  LOCKUP      WIRES
  -----
my_lockup1    my_wire1
```

SEE ALSO

- current_design(2)
- preview_dft(2)
- remove_scan_link(2)
- set_dont_touch(2)
- set_scan_configuration(2)
- set_scan_element(2)
- set_scan_link(2)
- set_scan_path(2)
- write(2)

report_scan_path

Displays scan paths and scan cells specified by the **set_scan_path** command and displays scan paths inserted by the **insert_dft** command.

SYNTAX

```
status report_scan_path
  [-view spec | existing_dft]
  [-chain chain_name | all]
  [-cell chain_name | all]
  [-test_mode mode_name_list | all]
```

Data Types

chain_name string
mode_name_list list

ARGUMENTS

-view spec | existing_dft

Specifies the type of scan information to report. Valid options are **spec** and **existing_dft**. Specifying the **spec** value shows only user-defined scan path information previously specified with the **set_scan_path** command. Specifying the **existing_dft** value shows only scan paths already inserted by the **insert_dft** command. The default behavior is to show both views.

-chain *chain_name* | all

When specified, reports only the summary scan chain information for the specified scan chains, or for all scan chains if **all** is specified. By default, both summary **-chain** and detailed **-cell** reports are generated.

-cell *chain_name* | all

When specified, reports only the detailed scan cell ordering information for the specified scan chains, or for all scan chains if **all** is specified. By default, both summary **-chain** and detailed **-cell** reports are generated.

-test_mode *mode_name_list* | all

Reports only on the specified test mode(s), or on all test modes if **all** is specified. By default, the tool reports on the current test mode. If no test modes were created, the default is to report on the Internal_scan mode.

DESCRIPTION

This command displays user-specified scan paths specified by the **set_scan_path** command, as well as the actual scan paths inserted by the **insert_dft** command.

The report contains three different section types.

- VIEW: Specification

This section contains user-specified scan path information specified with the **set_scan_path** command. This report can contain information on scan path segments, or scan-in, scan-out, and scan-enable signals along with any hookup pin information.

- VIEW: Existing DFT (-chain)

This section contains summary information about scan paths that currently exist in the design. The summary contains scan-in, scan-out, scan-enable, scan-clock, scan chain name, and scan chain length information in table form, but does not print the ordered scan cell names for each scan chain.

- VIEW: Existing DFT (-cell)

This section contains detailed information about the scan cell ordering of the scan paths that currently exist in the design.

By default, both the summary **-chain** and detailed **-cell** reports are generated. To restrict the report to only summary information, use the corresponding option. For example, to see only summary information for all scan chains, use the **-chain all** option.

EXAMPLES

The following is an example of **report_scan_path** in the first format reporting on scan paths in the **spec** view. In this report, the *mychain1* scan path has SI (scan-in), Q (scan-out), and two scan enables SE1 and SE2, which are both hooked up to pin u5/A.

```
prompt> report_scan_path -view spec -chain mychain1 -test_mode mymodeA
```

```
*****
```

```
Report : Scan path
```

```
Design : A
```

```
Version: 2003.12
```

```
Date : Thu Aug 14 10:46:10 2003
```

```
*****
```

```
=====
```

```
TEST MODE: mymodeA
```

```
VIEW : Specification
```

```
=====
```

Scan_path	ScanDataIn (h)	ScanDataOut (h)	ScanEnable (h)
mychain1	SI (-)	Q (-)	SE1 (u5/A) SE2 (u5/A)

The following is an example of **report_scan_path** in the second format reporting scan paths in the **existing_dft** view. In this report, the *mychain1* scan path is defined in the *mymodeA* test mode, and *mychain2* is defined in the *mymodeB* test mode. The *mychain2* scan chain has SI2 (scan-in), SO (scan-out), and SE2 & SE3 (scan-enable). Clocks MCLK1, MCLK2, and MCLK3 are master clocks for this scan chain, and SCLK is a slave clock.

```
prompt> report_scan_path -view existing_dft -chain all -test_mode all
```

```
*****
```

```
Report : Scan path
Design : A
Version: 2003.12
Date  : Thu Aug 14 10:46:10 2003
*****
```

```
=====
TEST MODE: mymodeA
VIEW  : Existing DFT
=====
Scan_path  Len  ScanDataIn  ScanDataOut  ScanEnable  MasterClock  SlaveClock
-----
mychain1   -   SI      Q      SE1      -      -
```

```
=====
TEST MODE: mymodeB
VIEW  : Existing DFT
=====
Scan_path  Len  ScanDataIn  ScanDataOut  ScanEnable  MasterClock  SlaveClock
-----
mychain2   5   SI2      SO      SE3      MCLK1      SCLK
              SE2      MCLK2      -
              -      MCLK3      -
```

The following is an example of **report_scan_path** in the third format reporting scan cells. In this report, 4 scan cells are reported from the *mychain1* scan chain:

```
prompt> report_scan_path -view existing_dft -cell mychain1 \
-test_mode Internal_scan
```

```
*****
Report : Scan path
Design : top
Version: 2003.12
Date  : Thu Aug 14 10:46:29 2003
*****
```

```
=====
TEST MODE: Internal_scan
VIEW  : Existing DFT
=====
Scan_path  Cell_#  Instance_name  Clocks
-----
mychain1   0      r0/r0/q_reg1/q_reg
           1      r0/r0/q_reg2/q_reg
           2      r0/r1/q_reg1/q_reg
           3      r0/r1/q_reg2/q_reg
```

SEE ALSO

```
current_test_mode(2)
report_dft_configuration(2)
```

```
report_dft_signal(2)
report_scan_configuration(2)
report_scan_path(2)
set_dft_configuration(2)
set_dft_signal(2)
set_scan_configuration(2)
set_scan_path(2)
```

report_scan_register_type

Displays test-related information about the current design.

SYNTAX

status **report_scan_register_type**

ARGUMENTS

The **report_scan_register_type** command has no arguments.

DESCRIPTION

This command displays the table of scan-register types specified using the **set_scan_register_type** command.

EXAMPLES

The following is an example of the **report_scan_register_type** command:

```
prompt> report_scan_register_type

*****
Report : test
Design : level1
Version: A-2007.12
Date   : Thu Dec 28 17:46:57 2007
*****
Scan register on design level1 has '2' cells :
  FD3S
  FD4S
-exact false
```

SEE ALSO

current_design(2)
insert_dft(2)
set_scan_register_type(2)

report_scan_replacement

Displays the scan-replacement table specified by the **set_scan_replacement** command.

SYNTAX

status **report_scan_replacement**

ARGUMENTS

The **report_scan_replacement** command has no arguments.

DESCRIPTION

This command produces a report showing the complete scan-replacement table specified using the **set_scan_replacement** command. The entries in this table are used by the **compile_scan**, **compile_ultra_scan**, and **insert_dft** commands to find scan equivalents for each instance of a non-scan flip-flop in the table.

To remove this table or entries in this table, use the **remove_scan_replacement** command.

EXAMPLES

The following is an example of the **report_scan_replacement** command:

```
prompt> report_scan_replacement
```

```
*****
```

```
Report : test
```

```
  -replacements
```

```
Design : level1
```

```
Version: A-2007.12
```

```
Date   : Thu Dec 28 18:18:38 2007
```

```
*****
```

```
Scan register on design level1 has '2' cells :
```

```
  FD3S
```

```
  FD4S
```

```
-exact false
```

Scan replacement table

Cell	LSSD	MUXD	CLOCKD_SCAN	AUX_CLOCKD_LSSD	COMB

FD1	_____	FD1S	_____	_____	_____
1					

SEE ALSO

- current_design(2)
- insert_dft(2)
- remove_scan_replacement(2)
- set_scan_replacement(2)

report_scan_skew_group

Reports scan skew groups defined by the **set_scan_skew_group** command.

SYNTAX

```
status report_scan_skew_group  
      [scan_skew_group_list]
```

Data Types

```
scan_skew_group_list  list
```

ARGUMENTS

scan_skew_group_list

When you specify a list of one or more scan skew group names, the command prints a detailed list of the elements in the specified scan skew groups. Wildcards are not supported.

By default, the command reports a summary of all scan skew groups defined, but does not include the individual elements in each scan skew group.

DESCRIPTION

This command displays the user-specified scan skew groups defined by the **set_scan_skew_group** command.

EXAMPLES

To report a summary of the scan skew groups defined, issue the command with no arguments:

```
prompt> report_scan_skew_group
```

```
*****
```

```
Report : Scan Skew Group
```

```
Design : top
```

```
Version: J-2014.09
```

```
Date   : Wed Jul 9 06:37:41 2014
```

```
*****
```

```
=====
```

```
TEST MODE: all_dft
VIEW   : Specification
```

```
=====
```

```
Scan_skew_group
```

```
-----
```

```
Number of user-defined scan skew groups: 3
```

```
G1
G2
G3
```

To report the detailed list of elements in particular scan skew groups, specify the list of groups to report:

```
prompt> report_scan_skew_group {G1 G2}
```

```
*****
```

```
Report : Scan Skew Group
```

```
Design : top
```

```
Version: J-2014.09
```

```
Date   : Wed Jul 9 06:37:53 2014
```

```
*****
```

```
=====
```

```
TEST MODE: all_dft
VIEW   : Specification
```

```
=====
```

```
Scan_skew_group
```

```
-----
```

```
G1      0      Z1P_reg[10]
G1      1      Z1P_reg[11]
G1      2      Z1P_reg[12]
```

```
G2      0      Z1P_reg[13]
G2      1      Z1P_reg[14]
G2      2      Z1P_reg[15]
```

SEE ALSO

```
remove_scan_skew_group(2)
```

```
set_scan_skew_group(2)
```

report_scan_state

Displays the scan state of the current design.

SYNTAX

status **report_scan_state**

ARGUMENTS

The **report_scan_state** command has no arguments.

DESCRIPTION

This command reports the current scan status description for the current design. The state of the design is either unknown, test_ready, or scan_existing.

EXAMPLES

The following example reports the scan state of the current design:

```
prompt> report_scan_state
```

```
*****  
Report : test  
       -state  
Design : top  
Version: V-2004.06-DFT  
Date   : Thu Mar 25 17:42:10 2004  
*****
```

```
Scan state      : scan cells replaced with loops
```

SEE ALSO

current_design(2)
insert_dft(2)
set_scan_state(2)

report_scan_suppress_toggling

Reports on the user specifications that were provided through the **set_scan_suppress_toggling** command in terms of the list of scan flip-flops to be gated by the **insert_dft** command.

SYNTAX

status **report_scan_suppress_toggling**

ARGUMENTS

The **report_scan_suppress_toggling** command has no arguments.

DESCRIPTION

The **report_scan_suppress_toggling** command reports on all of the existing specifications you provide through the **set_scan_suppress_toggling** command.

EXAMPLES

The following example shows how to issue the command:

```
prompt> report_scan_suppress_toggling
```

SEE ALSO

[insert_dft\(2\)](#)
[remove_scan_suppress_toggling\(2\)](#)
[set_scan_suppress_toggling\(2\)](#)

report_scenario_options

Reports the scenario options set by the **set_scenario_options** command.

SYNTAX

```
status report_scenario_options  
[-scenarios scenario_list]
```

ARGUMENTS

-scenarios *scenario_list*

Specifies a list of scenarios, active or inactive, to be reported. By default, the command reports scenario options for the current scenario.

DESCRIPTION

This command reports the values of scenario options set using the **set_scenario_options** command, for the current scenario or the scenarios listed using the **-scenarios** option. This command can report scenario options for inactive scenarios as well.

EXAMPLES

The following example uses **report_scenario_options** command to check the settings for the current scenario:

```
prompt> create_scenario MODE1  
Warning: Discarding all scenario specific information previously defined in  
this session. (UID-1008)  
Current scenario is: MODE1  
1  
  
prompt> set_scenario_options -setup false  
1  
  
prompt> report_scenarios_options  
  
*****  
Report : scenario options
```

Design : TEST03
Version: C-2009.06
Date : Wed Mar 19 14:57:08 2008

Scenario: MODE1 is active.

setup : false
hold : true
leakage_power : true
dynamic_power : true

SEE ALSO

all_scenarios(2)
create_scenario(2)
current_scenario(2)
report_scenarios(2)
set_scenario_options(2)

report_scenarios

Reports scenario setup information for a multi-scenario design.

SYNTAX

status **report_scenarios**

ARGUMENTS

The **report_scenarios** command has no arguments.

DESCRIPTION

The **report_scenarios** command reports scenario setup information for a multi-scenario design. It reports all defined scenarios, all active scenarios, the current scenario, the clock tree synthesis (CTS) scenario, and the scenario option settings, as well as other scenario-specific information for active scenarios. The scenario-specific information includes the library used for linking the scenario, the design operating condition for the scenario, and the TLUPlus files for the scenario.

The creation and usage of scenarios is available with Design Compiler Graphical.

Multicorner-Multimode Support

This command uses information from both active and inactive scenarios.

EXAMPLES

The following example shows sample output of the **report_scenarios** command.

```
prompt> report_scenarios
```

```
*****
```

```
Report : scenarios
```

```
Design : my_design
```

```
Scenario(s): wc_cworst_corner bc_cbest_corner
```

```
Version: D-2010.03-ICC-SP3
```

```
Date   : Thu Aug 19 10:34:13 2010
```

All scenarios (Total=2): wc_cworst_corner bc_cbest_corner
All Active scenarios (Total=2): wc_cworst_corner bc_cbest_corner
Current scenario : wc_cworst_corner
CTS scenario : not defined.

Scenario #0: wc_cworst_corner is active.
Scenario options: Setup Hold Leakage_power Dynamic_power
Has timing derate: Yes

Library(s) Used:
lib1_ccs (File: /usr/lib/lib1_ccs.db)
lib2_ccs (File: /usr/lib/lib2_ccs.db)
lib3_worst (File: /usr/lib/lib3_worst.db)
lib4_worst (File: /usr/lib/lib4_worst.db)
lib5_worst (File: /usr/lib/lib5_worst.db)

Operating condition(s) Used:
Analysis Type : on_chip_variation
Max Operating Condition: lib2_ccs:WORST
Max Process : 1.00
Max Voltage : 1.08
Max Temperature: 125.00
Min Operating Condition: lib2_ccs:WORST
Min Process : 1.00
Min Voltage : 1.08
Min Temperature: 125.00

TLU+ Files Used:
Max TLU+ file: /usr/tluplus/max_worst.tluplus
Min TLU+ file: /usr/tluplus/min_worst.tlupls
Tech2ITF mapping file: /usr/tluplus/map

Scenario #1: bc_cbest_corner is active.
Scenario options: Setup Hold Leakage_power Dynamic_power
Has timing derate: Yes

Library(s) Used:
lib1_ccst (File: /usr/lib/lib1_ccst.db)
lib2_ccst (File: /usr/lib/lib2_ccst.db)
lib3_best (File: /usr/lib/lib3_best.db)
lib4_best (File: /usr/lib/lib4_best.db)
lib5_best (File: /usr/lib/lib5_best.db)

Operating condition(s) Used:
Analysis Type : on_chip_variation
Max Operating Condition: lib1_ccst:BEST
Max Process : 1.00
Max Voltage : 1.32
Max Temperature: -40.00
Min Operating Condition: lib1_ccst:BEST
Min Process : 1.00
Min Voltage : 1.32
Min Temperature: -40.00

TLU+ Files Used:

Max TLU+ file: /usr/tluplus/max_best.tluplus

Min TLU+ file: /usr/tluplus/min_best.tlupls

Tech2ITF mapping file: /usr/tluplus/map

1

SEE ALSO

create_scenario(2)

set_scenario_options(2)

report_self_gating

Reports information about XOR Self-Gating performed by Power Compiler. This command is supported only in topographical mode.

SYNTAX

```
status report_self_gating  
    [-ungated]  
    [-nosplit]
```

ARGUMENTS

-ungated

Reports the names of all registers that are not self-gated, along with the reason for that.

-nosplit

Prevents line splitting and facilitates writing tools to extract information from the report output. Most design information is listed in fixed-width columns. If the information for a field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_self_gating** command reports information about self-gating cells and gated and ungated registers of the current design. To generate the report, **report_self_gating** uses self-gating attributes placed on the self-gating cells by the Power Compiler tool.

Any self-gating cell added by other means (for example, self-gating inferred by or instantiated in the RTL code) lacks these attributes and does not appear in the report.

Traditional clock-gating cells inserted by Power Compiler are also excluded from this report. Use **report_clock_gating** command to get information about them.

When the command is issued without options, the report lists the number of self-gating cells, the number of gated and ungated registers with percentages of the total, and the total number of registers.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows a report generated when the **report_self_gating** command is issued with no options. The report shows that out of 6 registers, 6 are self-gated and 0 are not. Also the multibit decomposition is shown when there are multibit registers in the design.

```
prompt> report_self_gating
```

```
*****
```

```
Report : self_gating
```

```
Design : top
```

```
Version: ...
```

```
Date : ...
```

```
*****
```

XOR Self Gating Summary

Number of self-gating cells	2
Number of self gated registers	6 (100.00%)
Number of registers not self-gated	0 (0.00%)
Total number of registers	6

Self Gating Multibit Decomposition

	Actual Count	Single-bit Equivalent
Number of Self Gated Registers		
1-bit	3	3
4-bit	3	12
Total	6	15
Number of Registers not Self-Gated		
1-bit	0	0
4-bit	0	0
Total	0	0
Total Number of Registers		
1-bit	3	3
4-bit	3	12
Total	6	15

The **-ungated** option displays additional information about ungated registers and the reason why they were not self-gated. Also, for some of these reasons there is a "What next" recommendation on how to fix it.

```
prompt> report_self_gating -ungated
```

```
prompt> report_self_gating -ungated
```

```
*****
```

```
Report : self_gating
        -ungated
Design : bot
Version: ...
Date   : ...
*****
```

Ungated Register Report

Ungated Register	Reason	What Next?
y_reg[9]	XOR Self gating creates negative slack on path Relax timing constraints on this path	
y_reg[8]	XOR Self gating creates negative slack on path Relax timing constraints on this path	
y_reg[7]	XOR Self gating creates negative slack on path Relax timing constraints on this path	
y_reg[6]	XOR Self gating creates negative slack on path Relax timing constraints on this path	
y_reg[5]	XOR Self gating creates negative slack on path Relax timing constraints on this path	
y_reg[4]	XOR Self gating creates negative slack on path Relax timing constraints on this path	
y_reg[3]	XOR Self gating creates negative slack on path Relax timing constraints on this path	
y_reg[2]	XOR Self gating creates negative slack on path Relax timing constraints on this path	
y_reg[1]	XOR Self gating creates negative slack on path Relax timing constraints on this path	
y_reg[0]	XOR Self gating creates negative slack on path Relax timing constraints on this path	

XOR Self Gating Summary

Number of self-gating cells	0
Number of self gated registers	0 (0.00%)
Number of registers not self-gated	10 (100.00%)
Total number of registers	10

SEE ALSO

```
compile_ultra(2)
all_self_gates(2)
report_clock_gating(2)
```

report_separate_process_options

Reports whether the tool uses separate UNIX processes for extraction, placement, and routing.

SYNTAX

status **report_separate_process_options**

DESCRIPTION

The **report_separate_process_options** command prints information about whether the tool uses separate UNIX processes for extraction, placement, and routing.

EXAMPLES

The following example shows a sample output of the **report_separate_process_options** command:

```
prompt> report_separate_process_options
```

```
*****
```

```
Report : separate_process_options
```

```
Version: E-2010.12-ICC
```

```
Date   : Tue Oct 5 16:22:44 2010
```

```
*****
```

```
Placer is in separate process mode
```

```
Router is in separate process mode
```

```
Extraction is in separate process mode
```

```
1
```

SEE ALSO

set_separate_process_options(2)

report_serialize_configuration

Displays options specified by the **set_serialize_configuration** command or the **reset_serialize_configuration** command.

SYNTAX

status **report_serialize_configuration**

ARGUMENTS

The **report_serialize_configuration** command has no arguments.

DESCRIPTION

This command displays options specified by the **set_serialize_configuration** command on the current design.

The command reports the following information:

- Number of deserializer inputs
- Number of deserializer outputs
- List of core instances and deserializer IP inputs
- List of core instances and serializer IP outputs
- Update stage enabled
- List of excluded clocks
- Serializer clock
- Name of the parallel mode

EXAMPLES

The following example shows a serialize configuration report:

```
prompt> report_serialize_configuration
```

```
*****
```

```
Report : Serializer Configuration
```

```
Design : CORE
```

```
Version: D-2010.03-SP3
```

```
Date  : Mon Jul 19 16:50:07 2010
```

```
*****
```

```
=====
```

```
TEST MODE: my_comp1
```

```
VIEW  : Specification
```

```
=====
```

Inputs	2
Outputs	2
update stage	True
Parallel mode	my_comp1
exclude clocks	Unspecified
serializer_clock	MY_SER_CLK

```
1
```

SEE ALSO

current_test_mode(2)

set_serialize_configuration(2)

reset_serialize_configuration(2)

report_size_only

Reports size_only cells in the current design along with their size_only types.

SYNTAX

```
status report_size_only  
  [-nosplit]  
  [-summary]  
  [cells]
```

Data Types

cells collection

ARGUMENTS

-nosplit

Prevents line splitting. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-summary

Displays only a summary report without detailed information about each size_only cell instance.

cells

Reports the specified size_only cells.

DESCRIPTION

The **report_size_only** command displays information about the size_only cells in the current design.

If the current instance is set, the report is generated for the design of that instance. Otherwise, the report is generated for the current design. Cells that do not have the **size_only** attribute are ignored.

You can specify a report for a collection of cells also. If you specify the **-summary** option, only the summary report is generated. If you specify both a collection of cells and **-summary**, the command generates a summary report using the cells in the collection without detailed information about each individual size_only cell in the collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of a size_only report:

```
prompt> report_size_only

*****
Report : size_only
Design : size1
Version: G-2012.06-SP1
Date   : Tue Aug 28 10:57:48 2012
*****

Descriptions of size_only Types:
timing      - Cell with timing constraint
user       - Cell has specific user-set size_only attribute

Cell              Types
-----
U2                timing
U3                user
-----

Summary:
Total 2 cells with size-only

*****
Distribution By Size-Only Type
*****
Type      Size-Only
-----
timing     1
user      1
1
```

SEE ALSO

```
list_size_only_types(2)
set_size_only(2)
set_compile_directives(2)
```

report_streaming_compression_configuration

Displays options specified by the **set_streaming_compression_configuration** command.

SYNTAX

status **report_streaming_compression_configuration**
[-test_mode *mode_name_list*]

Data Types

mode_name_list list

ARGUMENTS

-test_mode *mode_name_list*

Specifies the test mode(s) to report. If not specified, the tool reports on the *current_test_mode* that can be displayed by the **current_test_mode** command. If no test modes were created, the tool defaults to the default compression mode. If **all** is specified, the tool displays the scan configurations of all streaming compression test modes, including the default test mode (all_dft).

DESCRIPTION

This command displays options specified by the **set_streaming_compression_configuration** command on the current design.

The command reports the following information:

- Number of decompressor inputs
 - Number of compressor outputs
 - Compressed scan chain count
 - Maximum scan chain length
 - Base mode
-

EXAMPLES

The following is an example of a streaming compression configuration report:

```
prompt> report_streaming_compression_configuration
```

```
*****
```

```
Report : Streaming Compression Configuration
```

```
Design : top
```

```
Version: I-2013.12
```

```
Date   : Mon Oct 21 09:12:07 2013
```

```
*****
```

```
=====
```

```
TEST MODE: all_dft
```

```
VIEW    : Specification
```

```
=====
```

```
Inputs          2
```

```
Outputs         2
```

```
Chain Count     20
```

```
Maximum Chain Length      Unspecified
```

```
Base Mode        Unspecified
```

```
Clock            Unspecified
```

```
Min Power        False
```

```
1
```

SEE ALSO

`current_test_mode(2)`

`reset_streaming_compression_configuration(2)`

`set_streaming_compression_configuration(2)`

report_supply_net

Reports all supply nets in the current scope.

SYNTAX

```
status report_supply_net  
    [supply_net_name]  
    [-include_exception]
```

Data Types

supply_net_name string or list

ARGUMENTS

supply_net_name

Specifies the supply nets that are to be reported. If the supply net does not exist in the current scope, the command fails. If the option is not specified, all the supply nets in the current scope are reported.

-include_exception

Includes the exception connections of the supply net.

DESCRIPTION

The **report_supply_net** command reports detailed information about the supply nets in the current scope. The report for a supply net includes its full name, the scope in which it was created, its maximum and minimum voltages if set using the **set_voltage** command, its resolve type, the various supply states and the exception pins on the power net if the **-include_exception** option is specified. The names of the power domains to which the supply net belongs are also printed. The name of the power domain is suffixed with an asterisk (*) if the net is set as the domain supply net for that domain.

If the voltage of the supply net is not set, a ' U - ' is printed, indicating that the voltage is undefined.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example reports information about the supply nets *SN1* in the current scope:

```
prompt> report_supply_net
```

```
*****
```

```
Report : supply_net
```

```
Design : top
```

```
Version: A-2007.12-SP1
```

```
Date  : Wed Nov 21 03:15:21 2007
```

```
*****
```

```
-----
```

```
Supply Net      : SN1
```

```
Current Scope   : top
```

```
Operating Voltage  : -- Best Case -- -- Worst Case --  
                   (1.00)      (1.33)
```

```
Supply States    : active_state, off_state
```

```
Resolve Type     : unresolved
```

```
Exception Connections : I0/PWR, I1/PWR, PD0_INST/I1/PWR
```

```
Power Domain     : T A(*)
```

```
-----
```

```
1
```

SEE ALSO

```
create_power_domain(2)
```

```
create_supply_net(2)
```

```
connect_supply_net(2)
```

```
set_domain_supply_net(2)
```

report_supply_port

Reports information about the supply ports in the current scope.

SYNTAX

status **report_supply_port**
[*supply_port_name*]

Data Types

supply_port_name string or list

ARGUMENTS

supply_port_name

Specifies the supply port or list of ports to be reported. If a specified supply port does not exist in the current scope, the command fails. If this argument is not specified, the tool reports on all supply ports in the current scope.

DESCRIPTION

The **report_supply_port** command enables you to get detailed information about supply ports within the current scope. If *supply_port_name* is not specified, the tool reports on all supply ports within the current scope.

The report includes the full name of the supply port, the scope in which it is created, the direction, the supply states, and the supply net to which it is connected.

Supply ports that are present on switches can be printed by specifying the argument with the switch path name, slash (/), and supply port name as follows:

report_supply_port *switch_full_path_name/supply_port_name*

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports on all supply ports in the current scope:

```
prompt> report_supply_port
```

```
*****
Report : supply_port
Design : top
Version: A-2007.12-SP1
Date   : Wed Nov 21 03:14:25 2007
*****
```

```
-----
```

```
Supply Port    : SP1
Current Scope  : top
Direction     : Input
Supply net     : S_NET_1
Supply state names : active_state off_state
```

```
-----
```

```
1
```

The following example reports on a supply port on a power switch:

```
prompt> report_supply_port SWA/outport
```

```
*****
Report : supply_port
Design : top
Version: A-2007.12-SP3
Date   : Thu Mar 13 21:50:40 2008
*****
```

```
-----
```

```
Supply Port    : SWA/outport
Current Scope  : mid1
Direction     : Output
Supply net     : No supply net connected to this port
Supply state names : No supply states defined for the port
```

```
-----
```

```
1
```

SEE ALSO

create_supply_port(2)

report_synlib

Displays information about synthetic libraries.

SYNTAX

```
status report_synlib  
  library  
  [module_list]
```

Data Types

```
library    string  
module_list list
```

ARGUMENTS

library

Specifies the name of the library to be reported. This library must already be read into the tool or specified by the **search_path** variable.

module_list

Specifies a list of modules that are reported. The default is to report information about all modules in the synthetic library.

DESCRIPTION

The **report_synlib** command displays information about synthetic libraries.

The synthetic library report displays the following information in the following order:

- A list of the operators and their pins
- A list of the modules with their pins, parameters, attributes, implementations and bindings
- A list of the external implementations and bindings
- A list of DesignWare sub-blocks declared in the library

By default, all the modules are reported. If you specify the *module_list* option, the **report_synlib** command reports only the listed modules and the associated external implementations and bindings. The implementations are annotated according to the specified attributes.

If a module has external implementations in the specified library only, the **report_synlib** command searches for the definition of this module in the library list and displays the related information.

To identify the library objects that are added by the **update_lib** command, the library objects are marked with an asterisk (*). This is because the initial library is created by using the **read_lib** command.

To generate a report, load the specified library. To see the libraries that are currently loaded, use the **list-libraries** command. You can use the **search_path** variable to locate the library.

EXAMPLES

The following report displays the *DW01_add* and *DW01_sub* modules and their implementations in the *standard.sldb* library. At the end of the report, the super and fast implementations are marked with asterisks indicating that they are added by the **update_lib** command.

```
prompt> report_synlib standard.sldb { DW01_add DW01_sub }
```

```
*****
Report : library
Library: standard.sldb
Version: v3.1
Date   : Thu Jan 28 11:46:00 1993
*****
```

```
Library Type      : Synthetic
Tool Created      : v3.1
Date Created      : Oct. 29, 1992
Library Version   : 3.0a
```

Synthetic Modules:

Module	Pins	Dir	Width	Parameters

DW01_add	A	in	width	
	B	in	width	
	CI	in	1	
	SUM	out	width	
	CO	out	1	
				width = max(width('A'), width(B))
DW01_sub	A	in	width	
	B	in	width	
	CI	in	1	
	DIFF	out	width	
	CO	out	1	
				width = max(width('A'), width('B'))

Module Attributes:

```
Attributes:
u - dont_use
d - design_library
```

Module	Attributes
DW01_add	d = DW01
DW01_sub	d = DW01

Module Implementations:

Attributes/Parameters:

v - verify_only
 u - dont_use
 r - regular_licenses
 l - limited_licenses
 d - design_library
 s - priority_set_id
 p - priority
 leg - legal

Module	Implementations	Attributes/Parameters
DW01_add	cla	
	rpl	
	sim	v
DW01_sub	cla	
	rpl	
	sim	v

Module Bindings:

Module	Bindings	Operators	Pin Associations	Constraints	Unbound Oper Pin
DW01_add	b1	ADD_UNNS_OP	A, A		
		B, B			
		CI,"0"			
		SUM, Z			
	b2	ADD_TC_OP	A, A		
		B, B			
		CI,"0"			
		SUM, Z			
	b3	ADD_UNNS_OP	A, B		
		B, A			
		CI,"0"			
		SUM, Z			
DW01_sub	b1	SUB_UNNS_OP	A, A		
		B, B			
		CI,"0"			
		DIFF, Z			

```

b2    SUB_TC_OP  A, A
      B, B
      CI,"0"
      DIFF, Z

```

External Implementations:

Attributes/Parameters:

```

v - verify_only
u - dont_use
r - regular_licenses
l - limited_licenses
d - design_library
s - priority_set_id
p - priority
leg - legal

```

Module	Implementations	Attributes/Parameters
DW01_add	fast *	r = SynLib-ALU bar
DW01_sub	super *	l = SynLib-Eval p = "width > 8 ? 2 : 6"

EXAMPLES

The following command displays the *DW02_mult* module in the *dw_foundation.sldb* library. This module has only external implementations in the library. The definition for this module is in the *standard.sldb* library.

```
prompt> report_synlib dw_foundation.sldb DW02_mult
```

```
*****
```

```

Report : library
Library: dw_foundation.sldb
Version: F-2011.09-SP2
Date   : Sun Oct 23 21:12:57 2011
*****

```

```

Library Type      : Synthetic
Tool Created      : F-2011.09-SP2
Date Created      : 10.23.11
Library Version   : D-2010.03:F-2011.09-DWBB_201109.1:8031568a

```

Synthetic Modules:

Module
DW02_mult design_library: DW02 HDL parameter: A_width = width('A') HDL parameter: B_width = width('B') Parameter: PRODUCT_width = B_width + A_width module type: mult

Module Pins:

Attributes:

c - clock_pin

Module	Pins	Dir	Default Stall		Pin	Attributes
			Width	Value		
DW02_mult	A	in	A_width			
	B	in	B_width			
	TC	in	1			
	PRODUCT	out	PRODUCT_width			

Module Implementations:

Attributes/Parameters:

v - verify_only

V - verification implementation

u - dont_use

r - regular_licenses

l - limited_licenses

d - design_library

s - priority_set_id

p - priority

leg - legal

Module	Implementations	Attributes/Parameters
DW02_mult	verif	v
	csa	leg = "(A_width>=1) && (B_width>=1)"

Module Bindings:

Module	Binding
DW02_mult	b1 bound_operator: MULT_TC_OP Pin Associations (module, oper): A, A B, B TC,"1" PRODUCT, Z b2 bound_operator: MULT_TC_OP Pin Associations (module, oper): A, B B, A TC,"1" PRODUCT, Z b3 bound_operator: MULT_UNUS_OP Pin Associations (module, oper): A, A B, B TC,"0" PRODUCT, Z


```

b4    bound_operator: MULT_UNNS_OP
      Pin Associations (module, oper):
        A, B
        B, A
        TC,"0"
        PRODUCT, Z
    
```

External Implementations:

Attributes/Parameters:

```

v - verify_only
V - verification implementation
u - dont_use
r - regular_licenses
l - limited_licenses
d - design_library
s - priority_set_id
p - priority
leg - legal
    
```

Module	Implementations	Attributes/Parameters
DW02_mult	apparch	r = DesignWare d = DW02 leg = "(A_width>=1) && (B_width>=1)"
	nbw(replaced with pparch)	r = DesignWare d = DW02 leg = "(A_width>=1) && (B_width>=1)"
	pparch	r = DesignWare d = DW02 leg = "(A_width>=1) && (B_width>=1)"
	wall(replaced with pparch)	r = DesignWare d = DW02 leg = "(A_width>=1) && (B_width>=1)"

SEE ALSO

```

compare_lib(2)
list(2)
read(2)
read_lib(2)
report_lib(2)
write_lib(2)
    
```

report_target_library_subset

Reports target library subsets on the design.

SYNTAX

```
status report_target_library_subset  
  [-object_list cells]  
  [-top]  
  [-nosplit]
```

ARGUMENTS

-object_list *cells*

Specifies the cells on which the target library subset is to be reported.

-top

Specifies the target library subset being specified on the root design.

-nosplit

Specifies not to split lines when column fields overflow.

DESCRIPTION

This command reports the target library subset on the design based on the specified options. If neither option is specified, it reports the target library subset setting on the entire design.

Note that the command only reports the target library subset information that is explicitly set by the user. The command does not report the inherited target library subset information or the information from the **-milkyway_reflibs** option of the **set_target_library_subset** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the target library subset for the design and for block u1:

```
prompt> report_target_library_subset -top -object_list [get_cells u1]
```

SEE ALSO

- check_target_library_subset(2)
- compile(2)
- remove_target_library_subset(2)
- set_operating_conditions(2)
- set_target_library_subset(2)
- target_library(3)

report_test_assume

Displays the value set on the pins by the **set_test_assume** command.

SYNTAX

status **report_test_assume**

ARGUMENTS

The **report_test_assume** command has no arguments.

DESCRIPTION

This command displays the pins and the value they are set to by the **set_test_assume** command on the current design. It reports the test assume value of either 1 or 0.

EXAMPLES

The following is an example of the **report_test_assume** command:

```
prompt> report_test_assume
```

```
*****  
Report : test assume  
Design : sub  
Version: W-2004.12  
Date   : Thu Mar 31 15:48:46 2005  
*****
```

PIN NAME	Test Assume Value
-----	-----
ff1/CD	0
ff6/Q	1
ff2/CD	1

SEE ALSO

`remove_test_assume(2)`
`set_test_assume(2)`

report_test_model

Displays the test model information attached to a design.

SYNTAX

```
status report_test_model
[-design design_name]
```

Data Types

design_name string

ARGUMENTS

-design *design_name*

Specifies the design on which to report test model information.

DESCRIPTION

The **report_test_model** command displays the model information about a design. Specify the design with the **-design** option. If no design is specified, the model information of the current design is displayed.

Test modes are modes of operation of the scan design. The model information contains the list of test modes associated with that model and their type. If the design has no corresponding modes or model, the "No test modes" message appears.

EXAMPLES

The following example generates a test model report for the *M1* design:

```
prompt> report_test_model -design M1
```

```
*****
```

```
Report : test model
```

```
Design : M1
```

```
Version: 2001.08
```

```
Date   : Fri Oct 19 09:07:32 2001
```

Test Mode	Purpose	Spec	Impl	Model	Source
Internal_scan	InternalTest		Yes		
Mission_mode	Normal		Yes		
wrp_if	InternalTest		Yes		
wrp_of	ExternalTest		Yes		
wrp_safe	Isolate		Yes		

SEE ALSO

current_design(2)
insert_dft(2)

report_test_point_configuration

Displays options specified by the **set_test_point_configuration** command.

SYNTAX

status **report_test_point_configuration**

ARGUMENT

The **report_test_point_configuration** command has no arguments.

DESCRIPTION

This command displays the options specified by the **set_test_point_configuration** command on the current design.

EXAMPLES

The following are examples of **report_test_point_configuration**.

```
prompt> report_test_point_configuration
```

```
*****  
Report : Test Point configuration  
Design : des_unit  
Version: G-2012.06  
Date   : Wed Mar 14 15:24:16 2012  
*****
```

```
=====
```

TEST MODE: all_dft
VIEW : Specification

```
=====
```

Test Point Analysis Target:	Testability
Test Point Type:	Control

Maximum number of test points: 6
Maximum number of test points per scan register: 8
Clock type: Dominant

Test Point Type: Observe
Maximum number of test points: 1000
Maximum number of test points per scan register: 8
Clock type: Dominant
Power saving: Disable

1

prompt> **report_test_point_configuration**

Report : Test Point configuration
Design : des_unit
Version: G-2012.06
Date : Wed Mar 14 15:24:16 2012

=====
TEST MODE: all_dft
VIEW : Specification
=====

Test Point Analysis Target: Pattern Reduction

Test Point Type: Observe
Maximum number of test points: 7
Maximum number of test points per scan register: 3
Clock type: Dominant
Power saving: Disable

1

prompt> **report_test_point_configuration**

Report : Test Point configuration
Design : des_unit
Version: G-2012.06
Date : Wed Mar 14 15:24:16 2012

=====
TEST MODE: all_dft
VIEW : Specification
=====

Test point configuration is not specified.

1

SEE ALSO

reset_test_point_configuration(2)
set_test_point_configuration(2)

report_test_point_element

Displays options specified by the **set_test_point_element** command.

SYNTAX

```
status report_test_point_element  
      [list_of_design_pin_objects]  
      [-type test_point_type]
```

Data Types

```
list_of_design_pin_objects  list  
test_point_type           string
```

ARGUMENT

list_of_design_pin_objects

Specifies the list of design pin objects to which the **report_test_point_element** command applies.

-type *test_point_type*

Indicates the type to which the **report_test_point_element** command applies. The following values are valid:

```
control_0  
control_z0  
control_1  
control_z1  
control_01  
control_z01  
force_0  
force_z0  
force_1  
force_z1  
force_01  
force_z01  
observe
```

DESCRIPTION

This command displays the options specified by the **set_test_point_element** command on the current design.

EXAMPLES

The following is an example of **report_test_point_element**:

```
prompt> report_test_point_element
```

```
*****
```

```
Report : User-defined test point element
```

```
Design : UReset
```

```
Version: W-2004.12
```

```
Date   : Tue Aug 17 17:21:49 2004
```

```
*****
```

```
=====
```

```
TEST MODE: all_dft
```

```
VIEW      : Specification
```

```
=====
```

```
Type:                observe
```

```
Configuration ID:    0
```

```
Pins that need a test point:    CurrState_reg_0/Q CurrState_reg_1/Q
```

```
Control signal:      MY_TM (TestMode)
```

```
Clock signal:        MY_CLOCK ( MasterClock)
```

```
Test points per source or sink scan cell:    2
```

```
Scan for source or sink:    Enable
```

```
Power saving:        Enable
```

SEE ALSO

```
remove_test_point_element(2)
```

```
set_test_point_element(2)
```

report_testability_configuration

Displays options specified by the **set_testability_configuration** command.

SYNTAX

status **report_testability_configuration**
[-type *valid_type*]

ARGUMENT

DESCRIPTION

This command displays the options specified by the **set_testability_configuration** command for the current design.

SEE ALSO

reset_testability_configuration(2)
set_testability_configuration(2)

report_threshold_voltage_group

Reports the number, area, and percentage of cells for each threshold voltage group in the design. Also shows the number, area, and percentage of cells in user-specified low threshold voltage groups. Since the threshold voltage group attribute is specified on the library cells, hierarchical cells do not have a threshold voltage group. So, their usage is reported under the "undefined" threshold voltage group category.

SYNTAX

```
status report_threshold_voltage_group
      [cell_list]
      [-lvth_groups groups]
      [-nosplit]
      [-verbose]
```

Data Types

```
cell_list  list
groups     list
```

ARGUMENTS

cell_list

Specifies a list of cells to report the threshold voltage groups. If not specified, the report is generated on all cells of the current design.

-lvth_groups *groups*

Specifies the list of threshold voltage groups to be considered as low threshold voltage. The value of the *groups* argument is a list of threshold voltage group identifiers. Each identifier is a non-empty string that can contain alphanumeric characters and the underscore character ("_"). If you do not specify this option, the low threshold voltage groups specified earlier using the **set_multi_vth_constraint** command are used; otherwise, no low threshold voltage groups are reported.

-nosplit

Prevents line splitting and facilitates writing tools to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line, starting in the correct column.

-verbose

Lists all the cells belonging to each threshold voltage group. If this option is not specified, the default summary report shows only the number and percentage of cells.

DESCRIPTION

This command reports the number, area, and percentage of cells in each threshold voltage group for the specified list of cells. The report also shows separately, the number, area, and percentage of black box and non black box cells in each group.

The threshold voltage group is defined in the logic libraries with the **threshold_voltage_group** attribute on the cell, or with the **default_threshold_voltage_group** attribute on the library. You can also set the attributes on the objects using the **set_attribute** command. The attributes can be any string values, but the values must be related to the threshold voltage of the cell to be meaningful. Any cells in the design that have no threshold voltage group attribute are included in the "undefined" group. Hierarchical cells are an example of such cells. To report the threshold voltage group for a hierarchical cell, get the list of cells it contains using **[get_cells -hierarchical hierarchical_cell/*]** and pass it as the *cell_list* argument of the **report_threshold_voltage_group** command.

Use the **-lvth_groups** option to specify the list of threshold voltage groups that are to be considered as low threshold. When you specify this option, the specified low threshold voltage groups are marked by a letter "L" and it appears in the last column of the corresponding row in the report. The total number, area, and percentage of cells in all the low threshold voltage groups is also reported. Note that the low threshold voltage groups specified with the **-lvth_groups** option overrides any low threshold voltage groups specified in a previous command. Specifically, when you specify **-lvth_groups {}**, the command does not report any low threshold voltage groups even if you have previously defined low threshold voltage groups. However, if you do not specify the **-lvth_groups** option, the low threshold voltage groups that you specified by using the **set_multi_vth_constraint** command, are used to report the details of the low threshold voltage groups.

Use the **-verbose** option to list all the cells that belong to each threshold voltage group. Without this option, the command generates a summary that reports only the number, area, and percentage of cells.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the summary of cells in each threshold voltage group for the current design:

```
prompt> report_threshold_voltage_group -lvth_groups {lvt svt}
```

```
*****
```

Threshold Voltage Group Report

Threshold Voltage Group	All cells	Blackbox cells	Non-blackbox cells
hvt	34 (7.87%)	1 (50.00%)	33 (7.67%)
lvt	342 (79.17%)	1 (50.00%)	341 (79.30%) L
svt	56 (12.96%)	0 (0.00%)	56 (13.02%) L
Total	432 (100.00%)	2 (100.00%)	430 (100.00%)
Low vth groups	398 (92.13%)	1 (50.00%)	397 (92.33%)

Threshold Voltage Group	All cell area	Blackbox cell area	Non-blackbox cell area
hvt	185.36 (7.82%)	20.00 (66.67%)	165.36 (7.06%)

lvt	1975.60 (83.30%)	10.00 (33.33%)	1965.60 (83.94%) L
svt	210.60 (8.88%)	0.00 (0.00%)	210.60 (8.99%) L

Total	2371.56 (100.00%)	30.00 (100.00%)	2341.56 (100.00%)
Low vth groups	2186.20 (92.18%)	10.00 (33.33%)	2176.20 (92.94%)

Note: L indicates cells that belong to the low threshold voltage groups.

SEE ALSO

- compile_ultra(2)
- set_attribute(2)
- set_multi_vth_constraint(2)

report_timing

Displays timing information about a design.

SYNTAX

```
status report_timing
  [-to to_list
   | -rise_to rise_to_list
   | -fall_to fall_to_list]
  [-from from_list
   | -rise_from rise_from_list
   | -fall_from fall_from_list]
  [-through through_list
   | -rise_through rise_through_list
   | -fall_through fall_through_list]
  [-exclude exclude_list
   | -rise_exclude rise_exclude_list
   | -fall_exclude fall_exclude_list]
  [-path_type short | full | full_clock | full_clock_expanded | only | end]
  [-delay_type min | min_rise | min_fall | max | max_rise | max_fall]
  [-nworst paths_per_endpoint]
  [-max_paths max_path_count]
  [-input_pins]
  [-nets]
  [-transition_time]
  [-crosstalk_delta]
  [-capacitance]
  [-effective_capacitance]
  [-attributes]
  [-physical]
  [-slack_greater_than greater_slack_limit]
  [-slack_lesser_than lesser_slack_limit]
  [-lesser_path max_path_delay]
  [-greater_path min_path_delay]
  [-loops]
  [-enable_preset_clear_arcs]
  [-significant_digits digits]
  [-nosplit]
  [-sort_by group | slack]
  [-group group_name]
  [-trace_latch_borrow]
  [-derate]
  [-normalized_slack]
  [-scenarios scenario_list]
  [-temperature]
  [-voltage]
  [-unique_pins]
```

[-start_end_pair]
 [-variation]
 [-ignore_infeasible_paths]

Data Types

<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list
<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list
<i>exclude_list</i>	list
<i>rise_exclude_list</i>	list
<i>fall_exclude_list</i>	list
<i>paths_per_endpoint</i>	integer
<i>max_path_count</i>	integer
<i>greater_slack_limit</i>	float
<i>lesser_slack_limit</i>	float
<i>max_path_delay</i>	float
<i>min_path_delay</i>	float
<i>digits</i>	integer
<i>group_name</i>	string
<i>scenario_list</i>	list

ARGUMENTS

-to *to_list*

Reports only the paths to the named pins, ports, or clocks. If you do not specify the **-to** option, the default is to report the longest path to an output port if the design has no timing constraints. If the design has timing constraints, the default is to report the path with the worst slack within each path group if the **-group** option is not present. If the **-group** option is given, the default is to report the path with the worst slack within the group specified by the **-group** option.

-rise_to *rise_to_list*

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by the rising edge of the clock at clock source, taking into account any logical inversions along the clock path.

-fall_to *fall_to_list*

Same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by the falling edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-from *from_list*

Reports only the paths from the named pins, ports, or clocks. If you do not specify the **-from** option, the default is to report the longest path to an output port if the design has no timing constraints. If the design has timing constraints, the default is to report the path with the worst slack within each path group if the **-group** option is not present. If the **-group** option is given, the default is

to report the path with the worst slack within the group specified by the **-group** option.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by the rising edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by the falling edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-through *through_list*

Reports only paths that pass through the named pins, ports, or clocks. If you do not specify the **-through** option, the default is to report the longest path to an output port if the design has no timing constraints. If the design does have timing constraints, the default is to report the path with the worst slack within each path group if the **-group** option is not present. If the **-group** option is given, the default is to report the path with the worst slack within the group specified by the **-group** option.

If you specify the **-through** option only one time, the tool reports only the paths that travel through one or more of the objects in the list. You can specify the **-through** option more than one time in one command invocation. For a discussion of the use of multiple **-through** options, see the DESCRIPTION section.

-rise_through *rise_through_list*

Reports only paths with a rising transition at the specified objects. This option is similar to the **-through** option. You can specify the **-rise_through** option more than one time in a single command invocation. For a discussion of multiple **-through**, **-rise_through**, and **-fall_through** options, see the DESCRIPTION section.

-fall_through *fall_through_list*

Reports only paths with a falling transition at the specified objects. This option is similar to the **-through** option. You can specify the **-fall_through** option more than one time in a single command invocation. For a discussion of multiple **-through**, **-rise_through**, and **-fall_through** options, see the DESCRIPTION section.

-exclude *exclude_list*

Prevents reporting of all data paths from/through/to the named pins, ports, nets, and cell instances. Any data path that starts from, contains, or ends on a listed object is excluded from reporting. If you specify a cell instance, data paths that include any pin of that cell instance are excluded. This option has higher precedence than the **-from**, **-through**, **-to**, and similar options.

The exclusion does not apply to clock paths, even when the **-path_type full_clock** or **-path_type full_clock_expanded** option is used. It does not apply to borrowing path from the **-trace_latch_borrow** option.

This option is not applied to clock pins.

-rise_exclude *rise_exclude_list*

Same as the **-exclude** option, but applies only to paths rising at the named pins, ports, nets, and cell instances.

-fall_exclude *fall_exclude_list*

Same as the **-exclude** option, but applies only to paths falling at the named pins, ports, nets, and cell instances.

-path_type *short | full | full_clock | full_clock_expanded | only | end*

Specifies how to display the timing path. By default, the full path is displayed. The following values are valid:

- If **full_clock** is specified, the report is similar to the report you get with **full** but also reports the full clock paths for propagated clocks.

- If **full_clock_expanded** is specified, the report is similar to the report you get with **full_clock** but also reports the full clock path from the primary clock to the related generated clock source.
- If **short** is specified, only startpoints and endpoints are displayed.
- If **only** is specified, only the path is displayed without the accompanying required-time and slack calculation.
- If **end** is specified, the report has a column format that shows one line for each path, showing only the endpoint path total, required-time, and slack.

-delay_type min | min_rise | min_fall | max | max_rise | max_fall

Specifies the path type at the endpoint. The default is **max**.

-nworst *paths_per_endpoint*

Specifies the maximum number of paths to report per endpoint. The default is 1, which reports only the single worst path ending at a given endpoint.

-max_paths *max_path_count*

Specifies the number of paths to report per path group by default, or the number of paths to report for the whole design if the **timing_report_fast_mode** variable is set to true. The default is 1.

-input_pins

Shows input pins in the path report. The default is to show only output pins. This option also shows the delays of the nets connected to these pins.

-nets

Shows nets in the path report. The default is not to show nets. To show the delay for the nets, use the **-input_pins** option.

-transition_time

Shows the net transition time for each driving pin in the path report. The default is not to show the net transition time for each driving pin.

-crosstalk_delta

Shows the delta delay for each input pin in the path report. The default is not to show the delta delay for each input pin.

-capacitance

Causes the total (lumped) capacitance to be shown in the path report. The default is not to show capacitance. For each driver pin, the total capacitance driven by the driver is displayed in a column preceding both incremental path delay and transition time (specified with the **-transition_time** option). When the **-nets** option is specified, the capacitance is printed on the lines with nets instead of the lines with driver pins.

-effective_capacitance

Causes the effective capacitance for nets, calculated by the Arnoldi method, to be shown in the path report. For each driver pin, the worst effective capacitance among all of the driver arcs is displayed in a column labeled "Ceff". When the **-nets** option is used, the effective capacitance is shown in the lines reporting the nets, rather than the driver pins. Only capacitance values calculated by the Arnoldi method are shown in the "Ceff" column; where capacitance is calculated by another method, the "Ceff" column is left blank.

-attributes

Shows the attributes specified in the **timing_report_attributes** variable. The supported attributes are **dont_touch**, **dont_use**, **map_only**, **infeasible_paths** (for the Design Compiler and DC Explorer tools), **size_only** for cells, **dont_touch** for cells and nets, and **ideal_net** for nets.

-physical

Shows the locations of the pins and the capacitive loads for the pins and nets in the path report. The loads are displayed as a pair of values with the first value being the wire capacitance of the net and the second value being the total capacitance driven by the driver. If the pin location cannot be determined, the cell location is displayed, with the coordinates in microns.

-slack_greater_than *greater_slack_limit*

Specifies that only those paths with a slack greater than *greater_slack_limit* are to be reported. This option can be combined with **-slack_lesser_than** to report only those paths inside or outside a given slack range.

-slack_lesser_than *lesser_slack_limit*

Specifies that only those paths with a slack less than *lesser_slack_limit* are to be reported. This option can be combined with **-slack_greater_than** to report only those paths inside or outside a given slack range.

-lesser_path *max_path_delay*

Selects only those paths with a delay less than *max_path_delay*. Combine this option with the **-greater_path** option to select only those paths inside or outside a given delay range.

-greater_path *min_path_delay*

Selects only those paths with a delay greater than *min_path_delay*. Combine this option with the **-lesser_path** option to select only those paths inside or outside a given delay range.

-loops

Reports only the timing loops in the design.

-enable_preset_clear_arcs

Enables asynchronous timing arcs for this report. By default, asynchronous timing arcs are disabled during all timing verification. This option allows you to see the timing with these arcs enabled. Only the current report is affected.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are from 0 through 13. The default is 2. Using this option overrides the value set by the **report_default_significant_digits** variable.

-nosplit

Prevents line splitting and facilitates writing application to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-sort_by group | slack

Specifies the order in which the paths are reported. The default **-sort_by** key is **group**. By default, paths are sorted by costing groups. Within each group, the paths are ordered by slack. With **slack**, the paths are ordered by slack only.

-group *group_name*

Specifies the path group from which timing paths are selected for reporting, based on other specified options for reports. If the **-group** option is not specified, the reported paths are a subset of paths from all path groups. This option cannot be used with the **-loops** option.

-trace_latch_borrow

Controls the type of report generated for a path that starts at a transparent latch. If the path startpoint borrows from the previous stage, using this option causes the report to show the entire set of borrowing paths that lead up to the borrowing latch, starting with a nonborrowing path or a noninverting sequential loop. By default, the report shows only the last path in the sequence of

borrowing stages. Each stage is reported separately, showing the time borrowed and lent and the endpoints of the stage. The cumulative amount of borrowed time along a sequence of stages is not included in the report. The **-input_pins**, **-nets**, **-transition_time**, **-capacitance**, and **-significant_digits** options apply to every stage in the sequence of borrowing paths, but the remaining options (for example, **-from**) apply only to the last stage reported.

-derate

Prints timing derate values for each path element. By default, no derate value is printed. When this option is specified, the **-input_pins** and **-nets** options are automatically turned on. Input delay and ideal clock network latency is not derated.

-normalized_slack

Paths are gathered and sorted using normalized slack instead of slack. Normalized slack divides the slack by an idealized allowed propagation delay. In order to use this option, you must run the **update_timing** command with the **timing_enable_normalized_slack** variable set to true.

-scenarios *scenario_list*

Reports each scenario in the list separately, with up to **-max_paths** paths reported for each scenario. Inactive scenarios are skipped in the report. If this option is not given, only the current scenario is reported.

-temperature

Reports the operating condition temperature for each path element for multivoltage designs.

-voltage

Reports the operating condition voltage for each path element for multivoltage designs. When the operating condition for a macro, switch, or pad cell is inferred from other library cells, a tilde character (~) is displayed next to the voltage value in the timing report.

-unique_pins

Causes only the single worst timing path through any given sequence of pins to be reported. No other paths are reported for the same sequence of pins from startpoint to endpoint. For example, if the worst path starts with a rising edge at the first pin of a pin sequence, then paths starting with a falling edge are not reported for that sequence. For non-unate logic such as XOR gates, this greatly reduces the number of paths reported because of the large number of possible rising/falling edge combinations through the sequence of pins. Using this option can require longer runtimes when used with the **-nworst** *n* option because many paths must be analyzed to find the worst path through each pin sequence, but only the worst path is reported and counted toward the total number of requested paths.

-start_end_pair

Limits reporting to the single worst timing path per each combination of startpoint and endpoint found. No other paths are reported that have the same startpoint and same endpoint. For example, if the worst path starts at a register clock pin ff1/CK and ends at a register input pin ff5/D, the report omits all other less critical paths from ff1/CK to ff5/D. Using this option can require longer runtimes when used with the **-nworst** *n* option because many paths must be analyzed to find the worst path through each pin pair, but only the worst path in each case is reported and counted toward the total number of requested paths.

-variation

Reports the mean and standard deviation of the statistical time increment for each path element. This option works only when parametric on-chip variation (POCV) analysis has been enabled by setting the **timing_pocvm_enable_analysis** variable to true. By default, statistical parameters are not reported.

-ignore_infeasible_paths

Ignores all paths flagged as infeasible during the latest compilation.

DESCRIPTION

The **report_timing** command provides a report of timing information for the current design. By default, the **report_timing** command reports the single worst setup path in each clock group.

The command options let you specify the number of paths reported, the types of paths reported, and the amount of detail included in the report. You can restrict the scope of paths reported by startpoints, endpoints, and intermediate points by using the **-from**, **-to**, and **-through** options; and by slack or clock group by using the **-slack_lesser_than** and **-group** options.

The timing report starts by showing the primary command settings, operating conditions, path startpoint, path endpoint, path group name, and path timing check type (max for a setup check, min for a hold check, and so on).

A table in the report shows point-by-point accounting of the delays along the path from the startpoint to the endpoint. The default table has columns labeled Point, Incr, and Path. These columns list the points (cell pins) along the path, the incremental contribution to the delay at each point, and the cumulative delay up to that point, respectively. Hierarchical boundary crossings are listed as well, showing zero incremental delay at each crossing. You can optionally display net delays in the report by using the **-input_pins** option and net names by using the **-nets** option.

The symbols "r" and "f" in the Path column indicate the sense of the signal transition, either rising or falling, at that point in the path.

For a setup check, the path starts with the launch clock edge and ends at the data input of the capture device. The data arrival time shown in the table is the amount of elapsed time from the source of the launch clock edge to the arrival of data at the endpoint, taking into consideration the longest possible delays along the path.

Following this is the accounting for the required arrival time. The data required time shown in the table is the latest allowable arrival time for the data at the path endpoint, taking into account the nominal capture clock edge time, the clock network delay, the clock uncertainty, the least possible delay along the clock path, and the library setup time requirement for the capture device.

The slack value shown at the end of the report is the data required time minus the data arrival time. This represents the amount of time by which the timing constraint is met.

Back-annotations on path elements in the timing path are indicated by a character symbol in the Incr column. If the **-input_pins** option is used, each pin-to-pin delay spans either a net or cell. The symbol shown refers to the dominant annotation on each path element. (Certain annotations dominate others; for example, SDF takes precedence over back-annotated RC parasitics.)

Symbol	Annotation
-----	-----
^	Ideal network latency annotation
*	Back-annotation using SDF or preroute Elmore extraction
&	RC network back-annotation using Elmore delay calculation
#	Estimated delay for high-fanout net
z	Zero delay due to set_zero_interconnect_delay_mode
<none>	Wire-load model or none

You can use multiple **-through** options in a single command to specify paths that traverse multiple points in the design. The following example specifies paths beginning at A1, passing through B1, then through C1, and ending at D1:

```
prompt> report_timing -from A1 -through B1 -through C1 -to D1
```

If more than one object is specified within one **-through** option, the path can pass through any of the objects. The following example specifies paths beginning at A1, passing through either B1 or B2, then passing through either C1 or C2, and ending at D1:

```
prompt> report_timing -from A1 -through {B1 B2} -through {C1 C2} -to D1
```

Creation and usage of scenarios is available with Design Compiler Graphical.

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenarios** option. The only exception is the clock object, which only comes from current scenario.

EXAMPLES

The following example shows a timing report using only the default values:

```
prompt> report_timing
```

```
*****
Report : timing
  -path full
  -delay max
  -max_paths 1
Design : led
Version: v3.1a
Date   : Tue Apr 7 16:23:02 1992
*****
```

Operating Conditions:
Wire Loading Model Mode: top

Startpoint: c (input port)
Endpoint: z2 (output port)
Path Group: default
Path Type: max

Point	Incr	Path
input external delay	0.00	0.00 r
c (in)	0.00	0.00 r
u1/Z (IVA)	0.54	0.54 f
u0/Z (NR2)	1.20	1.74 r
u8/Z (IVA)	0.43	2.17 f
u7/Z (OR3)	1.24	3.41 f
z2 (out)	0.00	3.41 f
data arrival time		3.41
max_delay	0.00	0.00
output external delay	0.00	0.00
data required time		0.00
data required time		0.00
data arrival time		-3.41
slack (VIOLATED)		-3.41

The following example reports the longest path to Z1, without required-time and slack calculation:

```
prompt> report_timing -to z1 -nworst 2 -path_type only
```

```
*****
```



```

Report : timing
-path only
-delay max
-nworst 2
-max_paths 2
Design : led
Version: v3.1a
Date  : Tue Apr 7 16:52:43 1992
*****

```

```

Operating Conditions:
Wire Loading Model Mode: top

```

```

Startpoint: c (input port)
Endpoint: z1 (output port)
Path Group: default
Path Type: max

```

Point	Incr	Path
input external delay	0.00	0.00 f
c (in)	0.00	0.00 f
u1/Z (IVA)	0.60	0.60 r
u17/Z (AO7)	0.53	1.13 f
u18/Z (OR3)	1.24	2.37 f
z1 (out)	0.00	2.37 f
data arrival time		2.37

```

Startpoint: d (input port)
Endpoint: z1 (output port)
Path Group: default
Path Type: max

```

Point	Incr	Path
input external delay	0.00	0.00 f
d (in)	0.00	0.00 f
u20/Z (IVA)	0.53	0.53 r
u17/Z (AO7)	0.53	1.06 f
u18/Z (OR3)	1.24	2.30 f
z1 (out)	0.00	2.30 f
data arrival time		2.30

The following example reports the endpoint path delay, required time, and slack for each path:

```

prompt> report_timing -path_type end

```

```

*****
Report : timing
-path end
-delay max
Design : led
Version: v3.1a
Date  : Tue Apr 7 16:28:07 1992
*****

```

Operating Conditions:
Wire Loading Model Mode: top

Endpoint	Path Delay	Path Required	Slack
z2	3.41 f	0.00	-3.41
z3	3.03 f	0.00	-3.03
z4	2.77 f	0.00	-2.77
z6	2.69 r	0.00	-2.69
z0	2.59 f	0.00	-2.59
z1	2.37 f	0.00	-2.37
z5	2.26 f	0.00	-2.26

The following example reports the startpoints and endpoints of the path from a to z2:

prompt> **report_timing -from a -to z2 -path_type short**

Report : timing

-path short

-delay max

-max_paths 1

Design : led

Version: v3.1a

Date : Tue Apr 7 16:29:40 1992

Operating Conditions:

Wire Loading Model Mode: top

Startpoint: a (input port)

Endpoint: z2 (output port)

Path Group: default

Path Type: max

Point	Incr	Path
input external delay		0.00 0.00 f
a (in)	0.00	0.00 f
...		
z2 (out)	1.24	1.24 f
data arrival time		1.24
max_delay	0.00	0.00
output external delay	0.00	0.00
data required time		0.00
data required time		0.00
data arrival time		-1.24
slack (VIOLATED)		-1.24

The following example shows input pins in the report, in addition to the default values:

prompt> **report_timing -input_pins**

```
*****
Report : timing
  -path full
  -delay max
  -input_pins
  -max_paths 1
Design : led
Version: v3.1a
Date  : Tue Apr 7 16:32:28 1992
*****
```

Operating Conditions:
Wire Loading Model Mode: top

Startpoint: c (input port)
Endpoint: z2 (output port)
Path Group: default
Path Type: max

Point	Incr	Path

input external delay	0.00	0.00 r
c (in)	0.00	0.00 r
u1/A (IVA)	0.00	0.00 r
u1/Z (IVA)	0.54	0.54 f
u0/A (NR2)	0.00	0.54 f
u0/Z (NR2)	1.20	1.74 r
u8/A (IVA)	0.00	1.74 r
u8/Z (IVA)	0.43	2.17 f
u7/B (OR3)	0.00	2.17 f
u7/Z (OR3)	1.24	3.41 f
z2 (out)	0.00	3.41 f
data arrival time		3.41
max_delay	0.00	0.00
output external delay	0.00	0.00
data required time		0.00

data required time		0.00
data arrival time		-3.41

slack (VIOLATED)		-3.41

The following example shows input pins and nets in the report, and does not show required time and slack calculation:

prompt> **report_timing -input_pins -nets -path_type only**

```
*****
Report : timing
  -path only
  -delay max
  -input_pins
  -nets
  -max_paths 1
Design : led
Version: v3.1a
Date  : Tue Apr 7 16:34:20 1992
```

Operating Conditions:
Wire Loading Model Mode: top

Startpoint: c (input port)
Endpoint: z2 (output port)
Path Group: default
Path Type: max

Point	Incr	Path

input external delay	0.00	0.00 r
c (in)	0.00	0.00 r
c (net)	0.00	0.00 r
u1/A (IVA)	0.00	0.00 r
u1/Z (IVA)	0.54	0.54 f
cell24/n22 (net)	0.00	0.54 f
u0/A (NR2)	0.00	0.54 f
u0/Z (NR2)	1.20	1.74 r
cell24/n21 (net)	0.00	1.74 r
u8/A (IVA)	0.00	1.74 r
u8/Z (IVA)	0.43	2.17 f
cell24/n19 (net)	0.00	2.17 f
u7/B (OR3)	0.00	2.17 f
u7/Z (OR3)	1.24	3.41 f
z2 (net)	0.00	3.41 f
z2 (out)	0.00	3.41 f
data arrival time		3.41

SEE ALSO

[create_scenario\(2\)](#)
[current_scenario\(2\)](#)
[report_constraint\(2\)](#)
[set_operating_conditions\(2\)](#)
[set_timing_derate\(2\)](#)
[set_timing_ranges\(2\)](#)
[set_wire_load_min_block_size\(2\)](#)
[set_wire_load_mode\(2\)](#)
[set_wire_load_model\(2\)](#)
[set_wire_load_selection_group\(2\)](#)
[report_default_significant_digits\(3\)](#)
[timing_pocvm_enable_analysis\(3\)](#)
[timing_report_attributes\(3\)](#)
[timing_report_fast_mode\(3\)](#)

report_timing_derate

Reports timing derating factors previously set for the design or for specified objects using the **set_timing_derate** command.

SYNTAX

```
status report_timing_derate
  [-include_inherited]
  [-pocvm_guardband]
  [-pocvm_coefficient_scale_factor]
  object_list
  [-nosplit]
  [-scenarios scenario_list]
```

Data Types

```
object_list    list
scenario_list  list
```

ARGUMENTS

-include_inherited

Includes inherited derating factors in the report. The derating values reported are the ones used in the delay calculation. The name of the cell or design from which the derating is inherited is shown in the "Inherited" column of the report. When this option is not used, only the values set directly on the object are reported.

-pocvm_guardband

Reports the POCV guardband derating factors.

-pocvm_coefficient_scale_factor

Reports the POCV coefficient scale factors.

No more than one of the following three options can be used in the command: **-aocvm_guardband**, **-pocvm_guardband**, or **-pocvm_coefficient_scale_factor**.

object_list

Specifies the list of objects whose derating factors are to be reported. The objects in the list can be leaf cells, instances, library cells, and designs. If this option is not used in the command, it reports all objects in the current design with user-defined derating applied.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is

listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-scenarios *scenario_list*

Reports timing derating factors for the specified scenarios of a multi-scenario design. Inactive scenarios are skipped in the report. Each scenario is reported separately.

If you do not use this option, only the current scenario is reported.

DESCRIPTION

The **report_timing_derate** command reports timing derating factors previously set for the design or for specified objects using the **set_timing_derate** command

When the **-include_inherited** option is used, the command reports all derating factors that are used in delay calculation, including those inherited from other objects. By default, only the derating factors set directly on the object are reported.

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

In the following example, the first **report_timing_derate** command reports the derating factors that were set on the design; the second one reports the guardband derating factors that were set on the design.

```
prompt> set_timing_derate -early 0.95
prompt> set_timing_derate -late 1.05
prompt> set_timing_derate -early -aocvm_guardband 0.96
prompt> set_timing_derate -late -aocvm_guardband 1.06
prompt> report_timing_derate
*****
```

Report : timing derate

Design : top

Scenario(s): s1

Design	Derate	value	Scenario

top	clk_net_delay_min_early	0.95	s1
	clk_net_delay_min_late	1.05	s1
	clk_net_delay_max_early	0.95	s1
	clk_net_delay_max_late	1.05	s1
	data_net_delay_min_early	0.95	s1
	data_net_delay_min_late	1.05	s1
	data_net_delay_max_early	0.95	s1
	data_net_delay_max_late	1.05	s1

clk_cell_delay_min_early	0.95	s1
clk_cell_delay_min_late	1.05	s1
clk_cell_delay_max_early	0.95	s1
clk_cell_delay_max_late	1.05	s1
clk_cell_check_min_early	-	s1
clk_cell_check_min_late	-	s1
clk_cell_check_max_early	-	s1
clk_cell_check_max_late	-	s1
data_cell_delay_min_early	0.95	s1
data_cell_delay_min_late	1.05	s1
data_cell_delay_max_early	0.95	s1
data_cell_delay_max_late	1.05	s1
data_cell_check_min_early	-	s1
data_cell_check_min_late	-	s1
data_cell_check_max_early	-	s1
data_cell_check_max_late	-	s1

Cell	Derate	value	Scenario

Lib Cell	Derate	value

1
prompt> **report_timing_derate aocvm_guardband**

Report : timing derate

-aocvm_guardband

Design : top

Scenario(s): s1

Design	Derate	value	Scenario

top	clk_net_delay_min_early	0.96	s1
	clk_net_delay_min_late	1.06	s1
	clk_net_delay_max_early	0.96	s1
	clk_net_delay_max_late	1.06	s1
	data_net_delay_min_early	0.96	s1
	data_net_delay_min_late	1.06	s1
	data_net_delay_max_early	0.96	s1
	data_net_delay_max_late	1.06	s1
	clk_cell_delay_min_early	0.96	s1
	clk_cell_delay_min_late	1.06	s1
	clk_cell_delay_max_early	0.96	s1
	clk_cell_delay_max_late	1.06	s1
	clk_cell_check_min_early	-	s1
	clk_cell_check_min_late	-	s1
	clk_cell_check_max_early	-	s1
	clk_cell_check_max_late	-	s1
	data_cell_delay_min_early	0.96	s1
	data_cell_delay_min_late	1.06	s1
	data_cell_delay_max_early	0.96	s1
	data_cell_delay_max_late	1.06	s1
	data_cell_check_min_early	-	s1

```
data_cell_check_min_late      -    s1
data_cell_check_max_early     -    s1
data_cell_check_max_late      -    s1
```

Cell	Derate	value	Scenario

Lib Cell	Derate	value	

1

The following example reports all derating factors for the instance named s1:

```
prompt> report_timing_derate s1 -include_inherited
```

SEE ALSO

- report_timing(2)
- reset_timing_derate(2)
- set_timing_derate(2)

report_timing_requirements

Reports timing path requirements (user attributes) and related information.

SYNTAX

```
status report_timing_requirements
[-attributes]
[-ignored]
[-from from_list
| -rise_from rise_from_list
| -fall_from fall_from_list]
[-through through_list]
[-rise_through rise_through_list]
[-fall_through fall_through_list]
[-to to_list
| -rise_to rise_to_list
| -fall_to fall_to_list]
[-expanded]
[-nosplit]
```

Data Types

```
from_list      list
rise_from_list list
fall_from_list list
through_list  list
rise_through_list list
fall_through_list list
to_list       list
rise_to_list  list
fall_to_list  list
```

ARGUMENTS

-attributes

Lists the path delay attributes set on the design. If used with the **-from** or **-to** option, **-attributes** limits the report to the specified from and to objects. This option reports on attributes set by the **set_multicycle_path**, **set_false_path**, **set_max_delay**, and **set_min_delay** commands.

The **-attributes** option is the default when no other options are specified. Any *max_time_borrow* attributes on objects are printed.

-ignored

Lists path timing attributes set on the current design that are ignored. For example, a false path might be specified from port A to

port *Z1*, but there is no timing path between those points. Use **-from** or **-to** to limit the report to certain paths. Ignored attributes on objects are printed, such as **max_time_borrow** on a cell other than a level-sensitive latch.

-from *from_list*

Specifies a list of clocks, ports, cells, and pins in the current design. The **-from** option limits the report to information that was set with **-from** to a path-based command, such as **set_multicycle_path**. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-rise_from *rise_from_list*

Specifies a list of clocks, ports, cells, and pins in the current design, similar to the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from *fall_from_list*

Specifies a list of clocks, ports, cells, and pins in the current design, similar to the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through *through_list*

Specifies a list of ports, cells, and pins in the current design. The **-through** option limits the report to information that was set with **-through** to a path-based command, such as **set_multicycle_path**.

-rise_through *rise_through_list*

Specifies a list of ports, cells, and pins in the current design, similar to the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than once in a single command invocation as with the **-through** option.

-fall_through *fall_through_list*

Specifies a list of ports, cells, and pins in the current design, similar to the **-through** option, but applies only to paths with a falling transition at the specified objects. You can specify **-fall_through** more than once in a single command invocation as with the **-through** option.

-to *to_list*

Specifies a list of clocks, ports, cells, and pins in the current design. The **-to** option limits the report to information that was set with **-to** to a path-based command, such as **set_multicycle_path**. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-rise_to *rise_to_list*

Specifies a list of clocks, ports, cells, and pins in the current design, similar to the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-fall_to *fall_to_list*

Specifies a list of clocks, ports, cells, and pins in the current design, similar to the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-expanded

Expands compressed timing exceptions on the design. Compressed timing exceptions applied on the design are reported as

compressed exceptions by the **report_timing_requirements** command. Although every object in the from, to, and through list gets enumerated, the exception is still a compressed exception. With the **-expanded** switch, these exceptions are expanded so that each of the from, to, and through list references at the most one element.

It is considered best practice not to use this option, since it can create a very large report.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_timing_requirements** command produces a report showing information about timing requirements of the design.

Use the **reset_design** command to remove all timing attributes on the current design. To remove timing attributes from specified paths, use the **reset_path** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example lists all timing attributes set on the design:

```
prompt> report_timing_requirements
```

```
*****
```

```
Report : timing_requirements
```

```
-attributes
```

```
Design : counter
```

```
Version: v3.0
```

```
Date   : Thu Mar 19 19:28:32 1992
```

```
*****
```

From	Through	To	Setup	Hold
RESET	*	*	FALSE	FALSE
*	U1/Z, U5/A	CO	max=4.5	min=2.2
ffa	*	ffb	cycles=2	-

The following example lists all ignored timing attributes set on the design:

```
prompt> report_timing_requirements -ignored
```

```
*****
```

```
Report : timing_requirements
```

```
-ignored
```

```
Design : counter
```

Version: v3.0
Date : Mon Apr 20 19:03:38 1992

From	Through	To	Setup	Hold
QA	U1/Z	*	max=10	-
Object	Type	Attributes		
CLK	clock	max_time_borrow		

SEE ALSO

- current_design(2)
- set_false_path(2)
- set_max_delay(2)
- set_max_time_borrow(2)
- set_min_delay(2)
- set_multicycle_path(2)

report_tlu_plus_files

Reports the files used for TLUPlus extraction. This command is supported only in topographical mode.

SYNTAX

```
status report_tlu_plus_files
[-scenarios scenario_list]
```

Data Types

scenario_list list

ARGUMENTS

-scenarios *scenario_list*

Reports the TLUPlus files for the specified scenarios of a multi-scenario design. Non-existing scenarios are skipped in the report. Each scenario is reported separately. If you do not specify this option, only the current scenario is reported.

DESCRIPTION

This command reports the files used for virtual route and postroute extraction using TLUPlus.

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following example runs the **report_tlu_plus_files** command:

```
prompt> report_tlu_plus_files
```

SEE ALSO

`extract_rc(2)`

`set_tlu_plus_files(2)`

report_top_implementation_options

Reports the top-level design options for the linking and optimization of a block's interface logic using transparent interface optimization.

SYNTAX

status **report_top_implementation_options**

DESCRIPTION

The **report_top_implementation_options** command reports the top-level implementation options that are specified by using the **set_top_implementation_options** command for the blocks that belong to the current top-level design. The report is generated in a tabular form.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the top-level implementation options.

```
prompt> set_top_implementation_options -block_references blk1
prompt> set_top_implementation_options -block_references blk2 \
      -optimize_block_interface true
prompt> report_top_implementation_options
```

```
*****
Report   : Top implementation options
Version  : F-2011.09
Date     : Wed Jun 15 03:32:53 2011
*****
```

```
-----
Block    Optimize Optimize  Size Only Host
Reference Interface Shared Logic Mode  Options
-----
```

```
blk1    false    false    off    --
blk2    true     false    off    --
```

1

SEE ALSO

set_top_implementation_options(2)

report_track

Reports the routing tracks for a specified layer or for all layers.

SYNTAX

```
int report_track  
  [-layer layer]  
  [-dir X | Y]
```

Data Types

layer string

ARGUMENTS

-layer *layer*

Specifies the routing layer to use the routing tracks. You can use layer name, layer number, or a collection containing one layer object.

The default is to report the routing tracks on all layers.

-dir X | Y

Specifies the direction how routing tracks are placed. The valid values are **X** and **Y**; specify either. The default is to report routing tracks in both direction.

DESCRIPTION

This command reports a group of routing tracks for the routing layer.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports routing tracks on the floorplan.

```
prompt> report_track
*****

Report track
Design : design
Version: Y-2006.06-ICC-SP2
Date  : Thu Jul 6 00:34:03 2006
*****

Layer      Direction  Start    Tracks  Pitch    Attr
-----
Attributes :
  usr : User defined
  rt  : Route66 defined
  def : DEF defined

m2      Y    233.720    2861    0.560
m2      X    233.720    2649    0.560
m3      X    233.720    2649    0.560
m3      Y    233.720    2861    0.560
m2      Y    233.720    2861    0.560
m4      Y    233.720    2861    0.560
m4      X    233.720    2649    0.560
m3      X    233.720    2649    0.560
m5      X    233.720    2649    0.560
m5      Y    233.720    2861    0.560
m4      Y    233.720    2861    0.560
m6      Y    233.720    2861    0.560
m6      X    232.600    1327    1.120
m5      X    232.600    1327    1.120
m1      X    20.000     10     2.100    usr
1
```

```
prompt> report_track -layer m3 -dir horizontal
*****

Report track
Design : design
Version: Y-2006.06-ICC-SP2
Date  : Thu Jul 6 00:36:25 2006
*****

Layer      Direction  Start    Tracks  Pitch    Attr
-----
Attributes :
  usr : User defined
  rt  : Route66 defined
  def : DEF defined

m3      Y    233.720    2861    0.560
1
```

SEE ALSO

```
create_track(2)
remove_track(2)
```

report_transitive_fanin

Reports logic in the transitive fanin of specified sinks.

SYNTAX

```
status report_transitive_fanin  
-to sink_list  
[-nosplit]
```

Data Types

sink_list list

ARGUMENTS

-to *sink_list*

Specifies a list of sink pins, ports, or nets in the design. The transitive fanin of each sink in the *sink_list* is reported. If a net is specified, the effect is the same as listing all driver pins on the net. This argument is required.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_transitive_fanin** command produces a report showing the transitive fanin of specified sink pins, ports, or nets in the design. A pin is considered to be in the transitive fanin of a sink if there is a timing path through combinational logic from the pin to that sink. The fanin report stops at the clock pins of registers (sequential cells).

Use the **report_timing** command to show path delays in the design. Use the **report_transitive_fanout** command to see the fanout of a pin, port, or net.

If the *current_instance* is set, the report focuses on the fanin within the current instance. The report stops at the boundaries of the current instance, and paths outside the current instance are not reported. The report shows the hierarchical boundary pins on the current instance.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows the transitive fanin of two internal pins in the design:

```
prompt> report_transitive_fanin -to {u5/A u5/B}
```

```
*****
```

```
Report : transitive_fanin
```

```
Design : counter
```

```
Version: v3.1
```

```
Date  : Thu 1992
```

```
*****
```

The fanin network of sink u5/A is as follows:

Driver Pin	Load Pin	Type	Sense
u2/Z	u5/A	(net arc)	same

Load Pin	Driver Pin	Type	Sense
u2/A	u2/Z	IVA	opposite

Driver Pin	Load Pin	Type	Sense
b	u2/A	(net arc)	opposite

The fanin network of sink u5/B is as follows:

Driver Pin	Load Pin	Type	Sense
c	u5/B	(net arc)	same

SEE ALSO

```
current_design(2)
current_instance(2)
report_clock(2)
report_timing(2)
report_transitive_fanout(2)
```

report_transitive_fanout

Reports logic in the transitive fanout of specified sources.

SYNTAX

```
status report_transitive_fanout  
-clock_tree | -from source_list  
[-nosplit]
```

Data Types

source_list list

ARGUMENTS

-clock_tree

Indicates that all clock source pins and ports in the design are to be used as the list of sources. Clock sources are specified using the **create_clock** command. If there are no clocks, or if the clocks do not have sources, the report is empty. Use the **report_clock** command to list the sources for all clocks in the design. The **-clock_tree** option is a special form of the report that displays the clock trees (or "networks") in the design.

Either the **-clock_tree** or the **-from** option must be specified. The options are mutually exclusive, so specify only one.

-from *source_list*

Specifies a list of source pins, ports, and nets in the design. The transitive fanout of each source in the *source_list* is reported. If a net is specified, the effect is the same as listing all load pins on the net.

Either the **-clock_tree** or the **-from** option must be specified. The options are mutually exclusive, so specify only one.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_transitive_fanout** command produces a report showing the transitive fanout of specified source pins or ports in the design. A pin is considered to be in the transitive fanout of a source if there is a timing path through combinational logic from the source to that pin. The fanout report stops at the inputs to registers (sequential cells). The source pins or ports are specified with

either **-clock_tree** or **-from** *source_list*.

Use the **report_timing** command to show path delays in the design. Use the **report_transitive_fanin** command to see the fanin of a pin, port, or net.

If the *current_instance* is set, the report focuses on the fanout within the current instance. The report stops at the boundaries of the current instance, and does not report paths outside the current instance. The report also shows the hierarchical boundary pins on the current instance.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example lists the transitive fanout for all clock sources in the design:

```
prompt> report_transitive_fanout -clock_tree
```

```
*****
Report : transitive_fanout
        -clock_tree
Design : counter
Version: v3.0
Date   : Thu 1992
*****
```

The fanout network of source CLK is as follows:

Driver Pin	Load Pin	Type	Sense
CLK	ffd/CP	(net arc)	same
CLK	ffc/CP	(net arc)	same
CLK	ffb/CP	(net arc)	same
CLK	ffa/CP	(net arc)	same

The following example shows the transitive fanout of two internal pins in the design:

```
prompt> report_transitive_fanout -from {ffa/Q ffb/Q}
```

```
*****
Report : transitive_fanout
Design : counter
Version: F-2011.09
Date   : Wed Jul 27 17:21:06 2011
*****
```

The fanout network of source ffa/Q is as follows:

Driver Pin	Load Pin	Type	Sense
ffa/Q	QA/A	(net arc)	same

Load Pin	Driver Pin	Type	Sense

QA/A	QA/Z	IV	opposite
Driver Pin	Load Pin	Type	Sense

QA/Z	QA	(net arc)	opposite

The fanout network of source ffb/Q is as follows:

Driver Pin	Load Pin	Type	Sense

ffb/Q	QB/A	(net arc)	same
Load Pin	Driver Pin	Type	Sense

QB/A	QB/Z	IV	opposite
Driver Pin	Load Pin	Type	Sense

QB/Z	QB	(net arc)	opposite

SEE ALSO

- create_clock(2)
- current_design(2)
- current_instance(2)
- report_clock(2)
- report_timing(2)
- report_transitive_fanin(2)

report_units

Reports the units used for resistance, capacitance, timing, leakage power, current, and voltage in the flow. The units must be consistent with the main library units.

SYNTAX

string **report_units**

ARGUMENTS

The **report_units** command has no arguments.

DESCRIPTION

The **report_units** command displays the units used by the current design. To generate the report, the design should be loaded and linked to a library. The units are always reported in reference to the MKS units such as farad, amp, ohm, second, volt, and watt.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the **report_units** output:

```
*****
Report : units
Design : labf.mw
Version: Z-2007.03-ICC

Date  : Tue Aug 22 10:32:44 2006
*****
Units
-----
Time_unit      : 1e-09 Second(nS)
Capacitive_load_unit : 1e-12 Farad(pF)
```


Resistance_unit : 1000 Ohm(kohm)
Voltage_unit : 1 Volt
Power_unit : 1e-06 Watt(uW)
Current_unit : 1 Amp

SEE ALSO

set_units(2)

report_upf_cell_mismatch

Reports PM cells which are dirtily mapped by violating TLS, PVT or UPF constraint by using set_upf_cell_mismatch command.

SYNTAX

```
status report_upf_cell_mismatch
  -tls_violation
  -pvt_mismatch
  -no_unmapped
  [-cells cells]
  [-nosplit]
```

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

This command reports power management cells present in the netlist which were mapped by relaxations specified by the set_upf_cell_mismatch command ie. relaxations on target_library_subset, PVT matching and UPF constraint allowing for no unmapped power management cells in the final netlist.

All the options are optional. If just report_upf_cell_mismatch is specified without any options, then power management cells mapped by allowing any of the relaxations are reported.

EXAMPLES

The following example shows how to specify relaxations using report_upf_cell_mismatch command.

```
prompt> report_upf_cell_mismatch -nosplit
```

User specified command: set_upf_cell_mismatch -allow_pvt_mismatch

Violation:

tls - TLS violation
pvt - PVT mismatch
nu - No unmapped

Cell	Reference	PM Type	Violation

mid/snps_PD1__ISO_snps_q2_UPF_ISO	HS45_LS_ISOAND2X9 iso	pvt	
mid/snps_PD1__ISO_snps_q1_UPF_ISO	HS45_LS_ISOAND2X9 iso	pvt	

1			

SEE ALSO

set_upf_cell_mismatch(2)

report_use_test_model

Reports test model usage for each subdesign under the current design.

SYNTAX

status **report_use_test_model**

ARGUMENTS

The **report_use_test_model** command has no arguments.

DESCRIPTION

This command displays the list of subdesigns currently in dc_shell that are instantiated within the current design. For each subdesign, it indicates whether a test model or gates are used.

EXAMPLES

The following is an example of the **report_use_test_model** command:

```
prompt> report_use_test_model
```

SEE ALSO

use_test_model(2)

report_via_ladder_candidates

Reports via ladder candidates set by the **set_via_ladder_candidate** command.

SYNTAX

```
status report_via_ladder_candidates  
      lib_pin_list
```

Data Types

lib_pin_list collection

ARGUMENTS

DESCRIPTION

This command reports via ladder candidates on library pins.

EXAMPLES

The following example reports existing via ladder candidates on library pin "slow/TBUFIXL/Y"

```
prompt> report_via_ladder_candidates  
report_via_ladder_candidates  
library pin: slow/TBUFIXL/Y  
via ladder candidate: VL1_1_2 (400.000 220.000)  
                     VL1_2 (500.000 220.000)  
1
```

SEE ALSO

report_via_ladder_constraints

Reports via ladder constraints.

SYNTAX

```
status report_via_ladder_constraints  
-pins pins  
-output file_name
```

Data Types

```
pins      collection  
file_name string
```

ARGUMENTS

-pins *pins*

Collection of pins specified to report via ladder constraints.

-output *file_name*

Output file specified to writes the report.

DESCRIPTION

This command reports via ladder lists that were previously set by the **set_via_ladder_constraints** command.

A via ladder is a physical structure of connected vias and wires that is placed on top of a standard cell pin as a special connecting device. A via ladder can have, for example, two rows of via1, four columns of via2, and one row of via3 connected with a metal4 wire.

A named via ladder is defined by set of rules in the technology file.

This command also reports the via ladder added on pins during optimization.

EXAMPLES

The following command will report via ladders on all pins:

```
prompt> report_via_ladder_constraints
```

SEE ALSO

`set_via_ladder_constraints(2)`

`remove_via_ladder_constraints(2)`

report_via_ladder_rules

Reports via ladder rules set by the **set_via_ladder_rules** command.

SYNTAX

status **report_via_ladder_rules**

ARGUMENTS

The **report_via_ladder_rules** command has no arguments.

DESCRIPTION

This command reports existing via ladder rules.

This command returns 1 if succeeded, 0 otherwise.

EXAMPLES

The following example reports all via ladder constraints from the block.

```
prompt> report_via_ladder_rules
report_via_ladder_rules
master pin: lib/INVD8/ZN via ladder:{ VP2 }
master pin: lib/BUFFD10/Z via ladder:{ VP1 }
1
```

SEE ALSO

set_via_ladder_rules(2)
remove_via_ladder_rules(2)

report_via_rules

Displays `via_rule`'s information from the current library.

SYNTAX

```
string report_via_rules  
  [-verbose]  
  [-nosplit]  
  -all  
  via_rule_list
```

Data Types

via_rule_list string or collection

ARGUMENTS

-verbose

Outputs the various set properties names and their values for each of the `via_rule` along with default information. Name of the properties displayed is same as used in the technology file.

-nosplit

Prevents line splitting if the column overflows. Most information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

via_rule_list

Displays information only on the specified `via_rule` objects. *via_rule_list* contains either a collection of `via_rules` or a pattern that matches the `via_rule` names.

This option is mutually exclusive with the `-all` option, but you must specify one.

DESCRIPTION

This command displays information about `via_rules` in the current library. By default, the command produces a brief report containing `via_rule`'s name and if they are `via_ladder`.

EXAMPLES

Below example outputs the report for all via_rule objects.

```
prompt> report_via_rules -all
*****
Report : report_via_rules
Version:
Date :
*****
Name      Via ladder
-----
VR1       No
VL1       Yes
1
```

Below example outputs the verbose report for via_rule named VL1.

```
prompt> report_via_rules VL1 -verbose
*****
Report : report_via_rules
Version: L-2016.03-SP5-1-BETA
Date : Sun Oct 23 23:27:33 2016
*****
Name      Via ladder Property name      Value
-----
VL1       Yes      cutLayerNameTblSize      2
           cutLayerNameTbl      VIA1, VIA2
           cutNameTbl      cut1, cut2
           numCutRowsTbl      2, 2
           numCutsPerRowTbl    2, 2
1
```

SEE ALSO

```
get_via_rules(2)
remove_via_rules(2)
create_via_rule(2)
```

report_voltage_area

Reports the voltage areas in the design. This command is supported only in topographical mode.

SYNTAX

```
status report_voltage_area
  [-name list]
  [-nosplit]
  [-verbose]
  -all | patterns
```

Data Types

```
list      patterns
patterns  list
```

ARGUMENTS

-name *list*

Specifies the names of voltage areas to be reported.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-verbose

Reports both maximum and minimum operating conditions.

-all

Reports all voltage areas in the design. Voltage areas are created by using the **create_voltage_area** command for the hierarchical blocks.

This option is mutually exclusive with the *patterns* argument.

patterns

Specifies the list of hierarchical cells that each represent a voltage area.

This argument is mutually exclusive with the **-all** option.

DESCRIPTION

This command reports the user-specified voltage area constraints in the design, as specified by the **create_voltage_area** command. If you use the **-all** option, all voltage areas in the design are reported.

The report includes voltage area name, hierarchical blocks associated with voltage area, voltage area geometry, area of the voltage area, and utilization for the voltage area. The area is reported in square microns.

Voltage areas can physically be completely nested, that is, one voltage area lies completely inside another voltage area. When considering the area or utilization of the outside voltage area, the area of the inner voltage area is excluded.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the voltage area va1:

```
prompt> report_voltage_area -name va1
*****
Voltage area name va1
Block(s) :PPL_LBUS_CORE
Voltage area geometry : { 19.87 29.86 576.47 1148.89 }
Voltage area region Area : 622854.88
Voltage area Utilization : 0.66
Voltage Area Guard-band (X, Y) : (1, 1)
Voltage Area Max Operating Conditions : 1.16
Voltage Area Min Operating Conditions : 1.16
Voltage Area Max Operating Voltage : 1.160
Voltage Area Min Operating Voltage : 1.160
Voltage Area Max Operating Temperature : 125.000
Voltage Area Min Operating Temperature : -40.000
*****
1
```

SEE ALSO

create_voltage_area(2)
remove_voltage_area(2)

report_wire_load

Displays the characteristics of the wire-load models set on a design or in a library.

SYNTAX

```
status report_wire_load
  [-design design_name]
  [-name model_name]
  [-libraries]
  [-nosplit]
```

Data Types

<i>design_name</i>	string
<i>model_name</i>	string

ARGUMENTS

-design *design_name*

Specifies the design name for which to report wire-load models. The default is to report wire-load models set on the current design.

-name *model_name*

Specifies the name of the wire-load model to report. By default, the tool reports all of the wire-load models set on the specified design.

-libraries

Specifies to also report on wire-load models in the libraries used to link the current design.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the wire load model information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_wire_load** command displays characteristics of wire-load models set on a design, a cluster, or stored in a library. Wire-load models are set on a design or cluster with the **set_wire_load_model** command. Wire-loads models are created for designs

and clusters with the **create_wire_load** command.

The wire-load report displays where the model is stored under the Location field. This can be the current design, one of its subdesigns or clusters, or a library linking the current design. The Location is a design, subdesign, or cluster if the model was created with **create_wire_load**, otherwise the Location is a library.

If the wire-load model is in the wire_load format, a wire-load report displays the coefficients to calculate net resistance, capacitance, and area from the net length. The report displays the statistics on how the model was generated. Points indicates how many nets were used to calculate the average capacitance. Standard Deviation is the standard deviation indicating the accuracy of the length and average capacitance. A standard deviation close to zero indicates that all data points reside very close to the average. The standard deviation is the average delta between the model's capacitance for a fanout and each annotated capacitance for the same fanout. The weighted standard deviation is the average percentage of all standard deviation weighted with the number of points for each fanout. This measure characterizes the accuracy of the model created. The wire_load format is generated with the **create_wire_load** command when the compile_create_wire_load_tablevariable is set to true.

If **create_wire_load** is used to generate the wire load, and the **compile_create_wire_load_table** variable is set to **true**, the report for the wire-load table does not have the statistics information. The report has the length, capacitance, resistance, and area value listed for each fanout.

EXAMPLES

The following is an example of a wire-load report for the current design. The design has the wire_load format model. The **compile_create_wire_load_table** variable was set to **false** when the **create_wire_load** command was executed.

```
prompt> report_wire_load
```

```
*****
Report : wire loads
Design : counter
Version: v3.1
Date   : Mon Jan 25 10:47:24 1993
*****
```

```
Wire load model: counter_wl
Location      : counter
Resistance    : 0
Capacitance   : 1
Area          : 0
Slope         : 1.333
```

			Average	Standard	% Standard
Fanout	Length	Points	Cap	Deviation	Deviation
1	0.58	1565	0.58	0.01	0.99
2	1.46	168	1.46	0.04	2.78
3	2.40	64	2.40	0.12	4.80
4	3.33	56	3.33	0.15	4.59
5	4.03	30	4.03	0.24	5.88
6	5.31	23	5.31	0.35	6.58
7	6.58	14	6.58	0.78	11.87
8	7.86	11	7.86	0.94	11.91
9	14.80	8	14.80	2.97	20.07
10	18.93	2	18.93	9.93	52.47
11	23.06	3	23.06	8.75	37.94
12	27.19	1	27.19	12.85	47.28

13	31.32	4	31.32	6.82	21.77
16	31.74	6	31.74	5.84	18.40
17	32.89	16	32.89	3.16	9.62
18	34.04	6	34.04	4.51	13.26
19	35.18	6	35.18	3.67	10.43

 Weighted Average Standard Deviation: 2.08

The following is an example of a wire-load report for the current design. The design has the wire_load_table format model. The **compile_create_wire_load_table** variable was set to **true** when the **create_wire_load** command was executed.

prompt> **report_wire_load**

Report : wire loads

Design : TOP

Version: 1997.01-SI1

Date : Mon Sep 9 17:59:34 1996

Wire load model: U3/U7_wl

Location : CONVERTOR_1 (design)

Fanout	Length	Capacitance	Resistance	Area

1	0.28	0.28	.06	0
2	0.52	0.52	.09	0
3	0.71	0.71	.11	0

SEE ALSO

set_wire_load_mode(2)

set_wire_load_model(2)

set_wire_load_min_block_size(2)

set_wire_load_selection_group(2)

report_design(2)

report_wrapper_configuration

Reports the wrapper configuration of the current design.

SYNTAX

```
status report_wrapper_configuration
[-test_mode mode_name_list]
```

Data Types

```
mode_name_list list
```

ARGUMENTS

-test_mode mode_name_list

Specifies the test mode(s) to report. If not specified, the tool reports on the *current_test_mode* that can be displayed by the **current_test_mode** command. If no test modes were created, the tool defaults to the default compression mode. If **all** is specified, the tool displays the global wrapper configuration.

DESCRIPTION

This command reports the wrapper configuration in the following format:

Wrapper Structures	Status
-----	-----
Class:	<class_name>
Style:	<style_name>
Dedicated Cell Type:	<cell_type>
Shared Cell Type:	<cell_type>
Dedicated Cell Design Name:	<design_name>
Shared Cell Design Name:	<design_name>
Register IO Implementation:	<impl_name>
Use Dedicated Wrapper Clock:	<boolean>
Safe State:	<state>
Delay Test:	<boolean>

EXAMPLES

The following example reports the wrapper configuration with a dedicated cell type *WC_D1_S* and safe state 1:

```
prompt> set_wrapper_configuration class core_wrapper \  
-dedicated_cell_type WC_D1_S -safe_state 1  
1
```

```
prompt> report_wrapper_configuration
```

```
*****  
Report : Wrapper Configuration  
Design : M1  
Version: W-2004.12  
Date   : Thu Sep 16 09:46:34 2004  
*****
```

Wrapper Structures	Status
-----	-----
Class:	Core Wrapper
Style:	Dedicated Wrapper
Dedicated Cell Type:	WC_D1_S
Shared Cell Type:	WC_S1
Dedicated Cell Design Name:	
Shared Cell Design Name:	
Register IO Implementation:	Swap
Use Dedicated Wrapper Clock:	False
Safe State:	1
Delay Test:	False

```
1
```

SEE ALSO

```
insert_dft(2)  
preview_dft(2)  
reset_wrapper_configuration(2)  
set_wrapper_configuration(2)
```

report_write_lib_mode

Reports the **write_lib** mode of the tool, either enabled or disabled.

SYNTAX

status **report_write_lib_mode**

ARGUMENTS

None

DESCRIPTION

This command tells whether the **write_lib** mode of the tool is enabled or disabled. By default, the **write_lib** command is disabled. It can be enabled by using the **enable_write_lib_mode** command.

SEE ALSO

enable_write_lib_mode(2)
write_lib(2)

reset_autofix_configuration

Resets the global AutoFix configuration applied to the design for the specified fixing type.

SYNTAX

```
status reset_autofix_configuration  
-type fix_type
```

Data Types

fix_type string

ARGUMENTS

-type *fix_type*

Specifies the fixing type to report. The valid *fix_type* values are:

- all**
- clock**
- set**
- reset**
- xpropagation**
- internal_bus**
- external_bus**
- bidirectional**

DESCRIPTION

The **reset_autofix_configuration** command resets the global AutoFix configuration applied to the current design for the specified fixing type.

Note that this command only resets the specified fixing type to its default behavior. To disable an AutoFix fixing type completely, use the **set_dft_configuration** command.

EXAMPLES

The following example resets the global clock-fixing specification to the default behavior:

```
prompt> reset_autofix_configuration -type clock
```

SEE ALSO

`insert_dft(2)`
`report_autofix_configuration(2)`
`set_autofix_configuration(2)`

reset_autofix_element

Resets a local AutoFix configuration applied to design objects for the specified fixing type.

SYNTAX

```
status reset_autofix_element  
  list_of_design_objects  
  -type fix_type
```

Data Types

```
list_of_design_objects  list  
fix_type                string
```

ARGUMENTS

list_of_design_objects

Specifies the list of design objects to reset local fixing configurations for. Wildcards and collections are not supported.

Valid object types depend on the fixing type, as follows:

```
clock - cell (hierarchical and leaf)  
set - cell (hierarchical and leaf)  
reset - cell (hierarchical and leaf)  
xpropagation - cell (hierarchical and leaf)  
internal_bus - net  
external_bus - net  
bidirectional - port
```

-type *fix_type*

Specifies the fixing type to report. The valid *fix_type* values are:

```
all  
clock  
set  
reset  
xpropagation  
internal_bus  
external_bus  
bidirectional
```

DESCRIPTION

The **reset_autofix_element** command resets a local AutoFix configuration applied to design objects for the specified fixing type.

Note that this command only resets the specified fixing type to its default behavior. To disable an AutoFix fixing type completely, use the **set_dft_configuration** command.

EXAMPLES

The following example resets local clock-fixing specifications from cells *u0* and *u1*:

```
prompt> reset_autofix_configuration [list u0 u1] -type clock
```

SEE ALSO

insert_dft(2)
report_autofix_element(2)
set_autofix_element(2)

reset_bsd_configuration

Removes the IEEE 1149.1 or 1149.6 specifications from a boundary-scan design.

SYNTAX

status **reset_bsd_configuration**

ARGUMENTS

The **reset_bsd_configuration** command has no arguments.

DESCRIPTION

The **reset_bsd_configuration** command removes the IEEE 1149.1 or 1149.6 specifications from the current design.

EXAMPLES

The following is an example of the **reset_bsd_configuration** command:

```
prompt> reset_bsd_configuration
```

SEE ALSO

define_dft_design(2)
set_boundary_cell(2)
set_bsd_compliance(2)
set_bsd_instruction(2)
set_bsd_linkage_port(2)
set_scan_path(2)

reset_cell_mode

Resets the modes of the specified instances.

SYNTAX

```
status reset_cell_mode
      [instance_list]
```

Data Types

instance_list list

ARGUMENTS

instance_list

Specifies a list of instances for which modes are to be reset, which cancels the effects of any previous **set_cell_mode** commands for those instances. This enables all modes for the instances.

DESCRIPTION

The **reset_cell_mode** command resets the modes of the specified instances. If no instances are specified, all instances have their modes reset.

Applying this command to a cell that has no modes has no effect and does not generate any error or warning message.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example resets the modes of all instances whose name matches the description RAM*. You can use the **report_cell_mode** command to report the active mode of each instance that has modes. After executing **reset_cell_mode**, the read and write modes are active for both instances, as shown by the **report_cell_mode** command.

```
prompt> report_cell_mode Uram*
```



```
*****  
Report : mode  
Design : top_design  
*****
```

Cell	Mode(Group)	Status
Uram1/core(RAM2_core)	read(rw)	ENABLED
	write(rw)	disabled
Uram2/core(RAM2_core)	read(rw)	disabled
	write(rw)	ENABLED

```
prompt> reset_cell_mode RAM*
```

```
prompt> report_cell_mode RAM*
```

```
*****  
Report : mode  
Design : top_design  
*****
```

Cell	Mode(Group)	Status
Uram1/core(RAM2_core)	read(rw)	ENABLED
	write(rw)	ENABLED
Uram2/core(RAM2_core)	read(rw)	ENABLED
	write(rw)	ENABLED

SEE ALSO

report_cell_mode(2)
set_cell_mode(2)

reset_clock_gate_latency

Resets all clock-latency values previously specified for or applied to clock-gating cells.

SYNTAX

```
status reset_clock_gate_latency
[-clock clock_list]
```

Data Types

clock_list list

ARGUMENTS

-clock *clock_list*

Removes latency values only with respect to the specified clocks. If *clock_list* is not specified, latency is removed from all clocks that are currently defined.

DESCRIPTION

This command resets all clock-latency values previously specified for or applied to clock-gating cells. The following three effects are produced by issuing this command:

- Removes the attributes from the source ports of clocks where some clock-latency values may have been previously specified by using **set_clock_gate_latency** command.
- Removes all clock-latency information from clock-gating cells that already have annotated latency values. In this case, it does not differentiate between latencies set using the **set_clock_latency** command and those specified using the **set_clock_gate_latency** command.
- Removes the current fanout mode that was defined by using or not using the **-transitive_fanout** option from **set_clock_gate_latency** command.

Multicorner-Multimode Support

This command is scenario dependent and will affect the current scenario only. You must use this command on each scenario that already had a clock-gate latency setting to remove the settings. This is necessary if you wish to change the fanout mode of your **set_clock_gate_latency** constraint.

EXAMPLES

The following command removes clock-latency values from all clocks currently defined:

```
prompt> reset_clock_gate_latency
```

The following example shows how to reset the clock-gate latency constraint across different scenarios.

```
prompt> current_scenario sc1 \  
       reset_clock_gate_latency\  
       current_scenario sc2 \  
       reset_clock_gate_latency
```

SEE ALSO

```
apply_clock_gate_latency(2)  
remove_clock_latency(2)  
report_clock_gating(2)  
set_clock_gate_latency(2)  
set_clock_latency(2)
```

reset_design

Removes all user-specified objects and attributes from the current design, except those defined by using the **set_attribute** command and **UPF** commands.

SYNTAX

status **reset_design**

ARGUMENTS

The **reset_design** command has no arguments.

DESCRIPTION

The **reset_design** command removes all user-specified clocks, path groups, and attributes from the current design, except those defined by using the **set_attribute** command and **UPF** commands. This command returns zero if there is no current design.

WARNING: Executing the **reset_design** has wide-ranging consequences. This command removes all user-specified attributes on the design, with a result that is often equivalent to starting the design process from the beginning. If you want to reset only a few attributes, consider using the **remove_attribute** command. Alternatively, you might be able to remove selected attributes by using the commands that set them. Many attribute-setting commands, such as **set_cost_priority**, have a **-default** option that removes all attributes previously set by that command. See the appropriate man page to determine whether a specific **set_*** command has the **-default** option.

Note that when the **reset_design** command removes user-specified objects such as clocks, collections that contain those objects are implicitly deleted. For more information about collections, see the **collections** man page.

If any generated clocks derived from the library cell have been deleted by the **remove_generated_clock** command, the **reset_design** command restores them to their original undeleted state.

The **reset_design** command does not remove UPF constraints. To remove UPF constraints, use the **remove_upf** command instead.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following example resets the current design:

```
prompt> reset_design  
1
```

The following example illustrates how the **reset_design** command can implicitly delete collections:

```
prompt> set a [get_clocks]  
{clk}  
prompt> query_objects $a  
{clk}  
prompt> reset_design  
1  
prompt> query_objects $a  
Error: No such collection '_sel126' (SEL-001)
```

SEE ALSO

- current_design(2)
- remove_attribute(2)
- remove_clock(2)
- reset_path(2)
- attributes(3)
- remove_upf(2)

reset_dft_clock_controller

Resets the on-chip clocking controller specification for the current design.

SYNTAX

```
status reset_dft_clock_controller  
[-test_mode mode_name_list]
```

Data Types

mode_name_list list

ARGUMENTS

-test_mode *mode_name_list*

Specifies the test mode(s) to which the **reset_dft_clock_controller** command applies.

DESCRIPTION

This command resets the on-chip clocking controller specification for the current design.

Use the **set_dft_clock_controller** command to specify an on-chip clocking controller specification. Use the **report_dft_clock_controller** command to report the current on-chip clocking controller specification.

For more information, see the man page for the **set_dft_clock_controller** command.

EXAMPLES

In the following example, the current on-chip clocking controller specification is reset:

```
prompt> reset_dft_clock_controller
```

SEE ALSO

```
report_dft_clock_controller(2)
set_dft_clock_controller(2)
```

reset_dft_clock_gating_configuration

Resets the DFT clock-gating configuration for the current design.

SYNTAX

status **reset_dft_clock_gating_configuration**

ARGUMENTS

The **reset_dft_clock_gating_configuration** command takes no arguments.

DESCRIPTION

The **reset_dft_clock_gating_configuration** command resets the DFT clock-gating configuration for the current design.

EXAMPLES

In the following example, the command resets the DFT clock-gating configuration for the current design:

```
prompt> reset_dft_clock_gating_configuration
```

SEE ALSO

set_dft_clock_gating_configuration(2)
report_dft_clock_gating_configuration(2)

reset_dft_configuration

Resets the DFT configuration for the current design specified by the **set_dft_configuration** command.

SYNTAX

status **reset_dft_configuration**

ARGUMENTS

The **reset_dft_configuration** command has no arguments.

DESCRIPTION

The **reset_dft_configuration** command resets the DFT configuration for the current design.

EXAMPLES

In the following example, the command resets the current design to the default configuration:

```
prompt> reset_dft_configuration
```

SEE ALSO

insert_dft(2)
report_dft_configuration(2)
set_dft_configuration(2)

reset_dft_drc_rules

Resets the DFT DRC rule specifications to their default behaviors.

SYNTAX

```
status reset_dft_drc_rules  
  [-violation drc_list]  
  [-cell cell_list]
```

Data Types

```
drc_list    string  
cell_list   string
```

ARGUMENTS

-violation *drc_list*

Restricts the reset to the specified DRC violations.

-cell *cell_list*

Restricts the reset to the specified cells.

DESCRIPTION

The **reset_dft_drc_rules** command resets any modifications to how certain DRC rule violations affect DFT insertion that were applied by the **set_dft_drc_rules** command. The list of violation IDs supported by this command is:

- TEST-504 (warning) Cell %s has constant 0 value.
- TEST-505 (warning) Cell %s has constant 1 value.
- D17 (warning) D17 Clock input I of <type> S couldn't capture data.

By default, the command resets all DFT DRC specifications applied by the **set_dft_drc_rules** command. You can use the **-violation** and **-cell** options to restrict the scope of the reset.

Use the **set_dft_drc_rules** command to apply DFT DRC rule specifications. Use the **report_dft_drc_rules** command to display the DFT DRC rule specifications that have been applied.

EXAMPLES

The following example removes all DFT DRC specifications that apply to the TEST-504 and TEST-505 violations:

```
prompt> reset_dft_drc_rules -violation {TEST-504 TEST-505}
```

SEE ALSO

dft_drc(2)
insert_dft(2)
preview_dft(2)
report_dft_drc_rules(2)
set_dft_drc_rules(2)

reset_dft_insertion_configuration

Resets the DFT insertion configuration for the current design.

SYNTAX

status **reset_dft_insertion_configuration**

ARGUMENTS

The **reset_dft_insertion_configuration** command has no arguments.

DESCRIPTION

This command resets the existing insertion configuration to its defaults. The map effort default is medium, synthesis optimization is all, route_scan_enable is TRUE, route_scan_clock is TRUE, route_scan_serial is TRUE, preserve_design_name is FALSE, and the unscan option is FALSE. Use the **report_dft_insertion_configuration** command to display the current DFT insertion configuration of the design.

Use the **reset_dft_insertion_configuration** command to remove the current DFT insert configuration from the design.

EXAMPLES

The following example shows how to reset the insert DFT configuration:

```
prompt> reset_dft_insertion_configuration
```

SEE ALSO

insert_dft(2)
report_dft_insertion_configuration(2)
set_dft_insertion_configuration(2)

reset_ieee_1500_configuration

Resets the IEEE 1500 configuration for the current design.

SYNTAX

status **reset_ieee_1500_configuration**

ARGUMENTS

The **reset_ieee_1500_configuration** command has no arguments.

DESCRIPTION

The **reset_ieee_1500_configuration** command resets the IEEE 1500 configuration for the current design.

EXAMPLES

In the following example, the command resets the current design to the default IEEE 1500 insertion configuration:

```
prompt> reset_ieee_1500_configuration
```

SEE ALSO

report_ieee_1500_configuration(2)
set_ieee_1500_configuration(2)

reset_logicbist_configuration

Resets the LogicBIST compression configuration to the default configuration values for the current design.

SYNTAX

status **reset_logicbist_configuration**

ARGUMENT

The **reset_logicbist_configuration** command has no arguments.

DESCRIPTION

Use the **reset_logicbist_configuration** command to reset the LogicBIST compression configuration to the default configuration values for the design.

EXAMPLES

```
prompt> reset_logicbist_configuration
Accepted LogicBIST specification for design 'CORE'.
1
```

SEE ALSO

set_logicbist_configuration(2)
report_logicbist_configuration(2)
insert_dft(2)
preview_dft(2)

reset_mode

Resets the modes of the specified instances.

SYNTAX

```
status reset_mode  
[instance_list]
```

Data Types

instance_list list

ARGUMENTS

instance_list

Specifies a list of instances for which modes are to be reset. If using DCL, all modes except for the default are disabled. If using .lib, all modes of these instances are enabled. No error occurs if you specify to reset the modes on a cell that has no modes.

DESCRIPTION

The **reset_mode** command resets the modes of the specified instances. If no instances are specified, all instances have their modes reset. If using DCL, all modes except for the default are disabled. If using .lib, all modes of these instances are enabled. This is the default behavior when no modes are specified with the **set_mode** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example resets the modes of all instances whose name matches the description RAM*. You can use the **report_mode** command to report the active mode of each instance that have modes. Before executing **reset_mode**, instance *Uram1/core* was only in read mode, and instance *Uram2/core* was only in write mode. After executing **reset_mode**, the read and write modes are active for both instances, as shown by the **report_mode** command.

```
prompt> report_mode Uram*
```

```
*****
Report : mode
Design : top_design
*****
```

Cell	Mode(Group)	Status
Uram1/core(RAM2_core)	read(rw)	ENABLED
	write(rw)	
Uram2/core(RAM2_core)	read(rw)	disabled
	write(rw)	

prompt> **reset_mode RAM***

prompt> **report_mode RAM***

```
*****
Report : mode
Design : top_design
*****
```

Cell	Mode(Group)	Status

Uram1/core(RAM2_core)	read(rw)	ENABLED
	write(rw)	

Uram2/core(RAM2_core)	read(rw)	ENABLED
	write(rw)	

SEE ALSO

report_mode(2)
set_mode(2)

reset_path

Resets the specified paths to single cycle timing.

SYNTAX

```
status reset_path
  [-setup | -hold]
  [-rise | -fall]
  [-from from_list
   | -rise_from rise_from_list
   | -fall_from fall_from_list]
  [-through through_list
   | -rise_through rise_through_list
   | -fall_through fall_through_list]
  [-to to_list
   | -rise_to rise_to_list
   | -fall_to fall_to_list]
```

Data Types

<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list
<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list

ARGUMENTS

-setup

Specifies that the setup checking (maximum delay) is reset to single cycle behavior.

-hold

Specifies that the hold checking (minimum delay) is reset to single cycle behavior.

-rise

Specifies that the rising path delays are reset to single cycle behavior. The default is that both rising and falling delays are affected.

-fall

Specifies that falling path delays are set to single cycle behavior. The default is that both rising and falling delays are affected.

-from *from_list*

Specifies the names of clocks, pins, or cells to use to find path startpoints. If a specified object is a clock, all flip-flops, latches, and primary inputs related to that clock are used as path startpoints.

-rise_from *rise_from_list*

Specifies the names of clocks, pins, or cells to use to find path startpoints. This option is similar to the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from *fall_from_list*

Specifies the names of clocks, pins, or cells to use to find path startpoints. This option is similar to the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through *through_list*

Specifies a list of path throughpoints (port, pin, or leaf cell names) of the current design. The **reset_path** applies only to paths that pass through one of the points in the *through_list*. If more than one object is included, the objects must be enclosed either in quotation marks (") or in braces ({}). If you specify the **-through** option multiple times, the **reset_path** applies to paths that pass through a member of each *through_list* in the order the lists are given. In other words, the path must first pass through a member of the first *through_list*, then through a member of the second list, and so on for every through list specified. If you use the **-through** option in combination with the **-from** or **-to** options, the **reset_path** applies only if the **-from** or **-to** conditions are satisfied and the **-through** conditions are satisfied.

-rise_through *rise_through_list*

Specifies a list of path throughpoints (port, pin, or leaf cell names) of the current design. This option is similar to the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than once in a single command invocation as with the **-through** option.

-fall_through *fall_through_list*

Specifies a list of path throughpoints (port, pin, or leaf cell names) of the current design. This option is similar to the **-through** option, but applies only to paths with a falling transition at the specified objects. You can specify **-fall_through** more than once in a single command invocation as with the **-through** option.

-to *to_list*

Specifies the names of clocks, pins, or cells to use to find path endpoints. If a specified object is a clock, all flip-flops, latches, and primary outputs related to that clock are used as path endpoints.

-rise_to *rise_to_list*

Specifies the names of clocks, pins, or cells to use to find path endpoints. This option is similar to the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-fall_to *fall_to_list*

Specifies the names of clocks, pins, or cells to use to find path endpoints. This option is similar to the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the

paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

DESCRIPTION

This command specifies that designated timing paths in the current design are restored to the default single cycle behavior. The **reset_path** command is used to undo the effect of the **set_multicycle_path**, **set_false_path**, **set_max_delay**, and **set_min_delay** commands. Remove the attributes set by those commands with **reset_path**. Note that a general **reset_path** command removes the effect of matching general or specific point-to-point exception commands.

If you do not specify **-setup** or **-hold**, both are restored to the default behavior. The default is a setup relation of one cycle and a hold relation of 0 cycles. A hold value of 0 means that hold is checked one active edge before the setup data at the destination register.

If you specify **-setup**, only setup (maximum delay) checking is reset to the default behavior. If you specify **-hold**, only hold (minimum delay) checking is reset to the default behavior.

There is separate setup and hold information for rise and fall. In most cases, these are reset together. The **-rise** or **-fall** options are used to reset only one or the other.

To disable setup or hold calculations for paths, use the **set_false_path** command.

To see the nondefault path requirements on the design, use the **report_timing_requirements** command.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example resets the timing relation between *ff1/CP* and *ff2/D* to the default single cycle:

```
prompt> reset_path -from {ff1/CP} -to {ff2/D}
```

This example resets the rising delay to single cycle for all paths ending at *latch2d*:

```
prompt> reset_path -rise -to {latch2d}
```

This example shows how a specific and general point-to-point exception is set and then removed:

```
prompt> set_false_path 2 -from A -to Z
```

```
prompt> set_max_delay 15 -to Z
```

```
prompt> reset_path -to Z /* removes all exceptions to Z */
```

SEE ALSO

[current_design\(2\)](#)

```
report_constraint(2)
report_timing_requirements(2)
reset_design(2)
set_false_path(2)
set_max_delay(2)
set_min_delay(2)
set_multicycle_path(2)
```

reset_physical_constraints

Resets all current physical constraints.

SYNTAX

status **reset_physical_constraints**

ARGUMENTS

The **reset_physical_constraints** command has no arguments.

DESCRIPTION

The **reset_physical_constraints** command resets the physical constraints settings that were previously applied to the design. After this command is executed, the netlist does not have any physical information.

EXAMPLES

The following command resets the physical constraints:

```
prompt> reset_physical_constraints
```

SEE ALSO

extract_physical_constraints(2)
report_physical_constraints(2)
write_physical_constraints(2)

reset_pipeline_scan_data_configuration

Resets the pipeline scan data configuration specified by the **set_pipeline_scan_data_configuration** command.

SYNTAX

status **reset_pipeline_scan_data_configuration**

ARGUMENTS

The **reset_pipeline_scan_data_configuration** command has no arguments.

DESCRIPTION

This command resets the the pipeline scan data configuration for the current design.

EXAMPLES

The following example resets the pipeline scan data configuration of the current design to the default:

```
prompt> reset_pipeline_scan_data_configuration
```

SEE ALSO

[set_pipeline_scan_data_configuration\(2\)](#)

reset_scan_compression_configuration

Resets the scan compression configuration for the current design.

SYNTAX

status **reset_scan_compression_configuration**

ARGUMENTS

The **reset_scan_compression_configuration** command has no arguments.

DESCRIPTION

This command resets the current scan compression configuration on the design.

EXAMPLES

The following example shows the **reset_scan_compression_configuration** command:

```
prompt> reset_scan_compression_configuration  
Accepted scan compression specification for design 'des_unit'.  
1
```

SEE ALSO

insert_dft(2)
preview_dft(2)
report_scan_compression_configuration(2)
set_scan_compression_configuration(2)

reset_scan_configuration

Resets the scan configuration for the current design.

SYNTAX

```
status reset_scan_configuration  
[-test_mode mode_name_list]
```

Data Types

mode_name_list list

ARGUMENTS

-test_mode *mode_name_list*

Specifies the test mode(s) to which the **reset_scan_configuration** command applies. If this option is not specified, the default is to reset the current test mode. If you specify **all**, the global configuration is reset, but any mode-specific configurations are retained.

DESCRIPTION

The **reset_scan_configuration** command resets the current scan configuration from the design.

EXAMPLES

The following example shows how to reset the scan configuration:

```
prompt> reset_scan_configuration
```

SEE ALSO

```
insert_dft(2)  
report_scan_configuration(2)
```



```
set_scan_configuration(2)
```

reset_serialize_configuration

Resets the serialize configuration to the default serialize configuration values for the current design.

SYNTAX

status **reset_serialize_configuration**

ARGUMENT

The **reset_serialize_configuration** command has no arguments.

DESCRIPTION

Use the **reset_serialize_configuration** command to reset the serialize configuration to the default serialize configuration values for the design.

EXAMPLES

```
prompt> reset_serialize_configuration
Accepted serialize specification for design 'CORE'.
1
```

SEE ALSO

set_serialize_configuration(2)
report_serialize_configuration(2)
insert_dft(2)
preview_dft(2)

reset_streaming_compression_configuration

Resets the streaming scan compression configuration to the default configuration values for the current design.

SYNTAX

status **reset_streaming_compression_configuration**

ARGUMENT

The **reset_streaming_compression_configuration** command has no arguments.

DESCRIPTION

Use the **reset_streaming_compression_configuration** command to reset the streaming scan compression configuration to the default configuration values for the design.

EXAMPLES

```
prompt> reset_streaming_compression_configuration
Accepted streaming specification for design 'CORE'.
1
```

SEE ALSO

set_streaming_compression_configuration(2)
report_streaming_compression_configuration(2)
insert_dft(2)
preview_dft(2)

reset_switching_activity

Removes switching activity information from all objects in the current design.

SYNTAX

```
status reset_switching_activity  
[-verbose]  
[cell_list]
```

Data Types

cell_list list

ARGUMENTS

-verbose

Prints more information messages than are printed by default.

cell_list

Removes the switching activity information from all objects except top-level pins within the specified hierarchical cells.

DESCRIPTION

By default, this command enables you to remove the annotated switching activity throughout the entire design. However, if the *cell_list* option is provided, then this command removes the annotated switching activity of all objects within the specified cells except for their top-level pins.

To remove the annotated switching activity for specific nets, ports, and pins, use the **set_switching_activity** command. See the **set_switching_activity** man page for additional details.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes the switching activity information from the *TOP_MULT* design:

```
prompt> current_design TOP_MULT
```

```
prompt> reset_switching_activity
```

SEE ALSO

`read_saif(2)`

`set_switching_activity(2)`

reset_test_mode

Resets the test mode for the current design.

SYNTAX

status **reset_test_mode**

ARGUMENTS

The **reset_test_mode** command has no arguments.

DESCRIPTION

The **reset_test_mode** command resets the current test mode of the current design to the default test mode.

EXAMPLES

The following example resets the current test mode:

```
prompt> reset_test_mode  
Current test mode is reset to default  
1
```

SEE ALSO

current_test_mode(2)
define_test_mode(2)
remove_test_mode(2)

reset_test_point_configuration

Resets the automatic test point insertion configuration for the current design.

SYNTAX

status **reset_test_point_configuration**

ARGUMENTS

The **reset_test_point_configuration** command has no arguments.

DESCRIPTION

The **reset_test_point_configuration** command resets all automatic test point configuration options, set by the **set_test_point_configuration** command, to their default values.

If automatic test point insertion is enabled with the **set_dft_configuration -test_points enable** command, the default is to perform pattern reduction test point insertion. The **reset_test_point_configuration** command does not disable automatic test point insertion. To disable automatic test point insertion, use the following command:

```
prompt> set_dft_configuration -test_points disable
```

EXAMPLES

The following example shows how to reset the test point configuration to the default specifications:

```
prompt> reset_test_point_configuration
```

SEE ALSO

insert_dft(2)
report_test_point_configuration(2)
set_test_point_configuration(2)

reset_testability_configuration

Resets the testability configuration for the current design.

SYNTAX

status **reset_testability_configuration**
[-type *type*]

ARGUMENTS

DESCRIPTION

The **reset_testability_configuration** command resets the current testability configuration for the current design.

SEE ALSO

insert_dft(2)
report_testability_configuration(2)
set_testability_configuration(2)

reset_timing_derate

Removes all derating factors set on the current design and libraries.

SYNTAX

```
string reset_timing_derate  
    [-pocvm_guardband]  
    [-pocvm_coefficient_scale_factor]
```

ARGUMENTS

-pocvm_guardband

Removes only the POCV guardband derating factors previously set with the **-pocvm_guardband** option of the **set_timing_derate** command.

-pocvm_coefficient_scale_factor

Removes only the POCV coefficient scaling factors previously set with the **-pocvm_coefficient_scale_factor** option of the **set_timing_derate** command.

DESCRIPTION

This command removes all timing derating factors from the current design or libraries previously set with the **set_timing_derate** command. After the command has been run, the result is the same as if timing derating factors had never been set.

Use the **-aocvm_guardband** option to remove only the AOCV guardband derating factors.

Use the **-pocvm_guardband** or **-pocvm_coefficient_scale_factor** option to remove only those specific types of POCV derating factors.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example removes derating factors for all of the objects in the current design or libraries.

```
prompt> reset_timing_derate
```

The following example removes only the guardband derating factors from the objects in the current design and libraries. After that, the **report_timing_derate -aocvm_guardband** command shows no guardband derating values.

```
prompt> reset_timing_derate -aocvm_guardband
prompt> report_timing_derate -aocvm_guardband
```

```
*****
Report : timing derate
        -aocvm_guardband
Design : top
Scenario(s): s1
*****
```

1

SEE ALSO

report_timing(2)
report_timing_derate(2)
set_timing_derate(2)

reset_wrapper_configuration

Resets the wrapper configuration for the current design.

SYNTAX

```
status reset_wrapper_configuration
[-test_mode mode_name]
```

Data Types

mode_name string

ARGUMENTS

-test_mode *mode_name*

Specifies the test mode to which the **reset_wrapper_configuration** command applies. The default is **all**, which resets the global wrapper configuration, but keeps any mode-specific configurations.

DESCRIPTION

This command resets the wrapper configuration with the following values:

Wrapper Structures	Status
-----	-----
Class:	Core Wrapper
Style:	Dedicated Wrapper
Dedicated Cell Type:	WC_D1
Shared Cell Type:	WC_S1
Dedicated Cell Design Name:	
Shared Cell Design Name:	
Register IO Implementation:	Swap
Use Dedicated Wrapper Clock:	False
Safe State:	None
Delay Test:	False

EXAMPLES

The following example resets the wrapper configuration for the current design. A report command follows that shows the wrapper configuration values after the reset.

```
prompt> reset_wrapper_configuration  
1
```

```
prompt> report_wrapper_configuration
```

```
*****  
Report : Wrapper Configuration  
Design : M1  
Version: A-2007.12  
Date   : Fri Dec 28 09:46:28 2007  
*****
```

Wrapper Structures	Status
-----	-----
Class:	Core Wrapper
Style:	Dedicated Wrapper
Dedicated Cell Type:	WC_D1
Shared Cell Type:	WC_S1
Dedicated Cell Design Name:	
Shared Cell Design Name:	
Register IO Implementation:	Swap
Use Dedicated Wrapper Clock:	False
Safe State:	None
Delay Test:	False

```
1
```

SEE ALSO

```
insert_dft(2)  
preview_dft(2)  
report_wrapper_configuration(2)  
set_wrapper_configuration(2)
```

resize_polygon

Pushes the edges of the specified polygon inward or outward by the specified direction and distance.

SYNTAX

```
list resize_polygon  
  polygons  
  -size size  
  -size_left size  
  -size_right size  
  -size_bottom size  
  -size_top size
```

Data Types

polygons polygon list or collection
size double

ARGUMENTS

polygons

Specifies the list or collection of input polygons whose edges are pushed inward or outward by the specified direction and distance. You can specify a single polygon or multiple polygons.

To specify a polygon as a list, enter a sequence of X-Y coordinate pairs that represent the successive vertices of the polygon, using the following format:

```
{{x1 y1} {x2 y2} ... {xN yN} {x1 y1}}
```

The coordinate unit size is specified in the technology file; typically it is microns.

Polygons must be rectilinear, so the X coordinate of each data point must match the X coordinate of a neighboring data point, and the Y coordinate must match the Y coordinate of the other neighboring data point. The last point in the list must be the same as the first.

Instead of specifying lists of coordinates, you can specify a polygon collection. In that case, the output of the command is also a collection.

-size *size*

Specifies the distance that the edges of the polygon are moved. Enter a negative value to push the edges inward or a positive number to push the edges outward. The coordinate unit is specified in the technology file; typically it is microns.

-size_left *size*

Specifies the distance that the left-facing edges are moved. Enter a negative value to push these edges inward or a positive number to push these edges outward.

-size_right size

Specifies the distance that the right-facing edges are moved.

-size_bottom size

Specifies the distance that the bottom-facing edges are moved.

-size_top size

Specifies the distance that the top-facing edges are moved.

DESCRIPTION

This command pushes the edges of the specified polygon inward or outward by the specified distance and returns the list of polygons whose edges have been moved. Polygons can be represented as lists of coordinates or as a collection. Before using this command, the Milkyway design library must be open.

The **-size** option resize each polygon in all four directions, whereas the **-size_left**, **-size_right**, **-size_bottom**, and **-size_top** options resize each polygon only in the specified direction. You can use any combination of these four options together at the same time, but not with **-size** option.

If you specify a positive size value, the edges of the polygon are pushed outward to increase the size of the polygon, which is called oversizing the polygon. For example, if you use **-size_left 10** to size a polygon, the left edges of the polygon are moved outward (left) by 10 units. Oversizing fills any gaps that are less than two times the distance unit.

When you oversize a polygon, the command returns a single polygon. You can directly pass this result to another polygon command such as **convert_from_polygons**, **get_polygon_area**, **resize_polygon**, or **compute_polygons**.

If you specify a negative size value, the edges of the polygon are pushed inward and the size of the polygon is reduced, which is called undersizing the polygon. For example, if you use **-size_left -10** to size a polygon, the left edges of the polygon are moved inward (right) by 10 units.

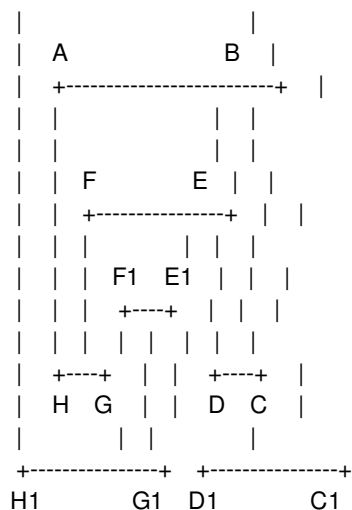
When you undersize a polygon, the result can be a disjoint rectilinear region that is represented by more than one polygon. Because the polygon commands take a single polygon as input, you cannot directly pass the result as a parameter to another polygon command. Instead, you must use a Tcl list command, such as the **foreach** or **lindex** command, to extract each polygon from the returned list, and then pass each polygon to the polygon command.

If you use more than one of the **-size_left**, **-size_right**, **-size_bottom**, or **-size_top** options at the same time, the size values should be consistently all positive or all negative, so that either oversizing or undersizing is performed in all directions that are being modified.

EXAMPLES

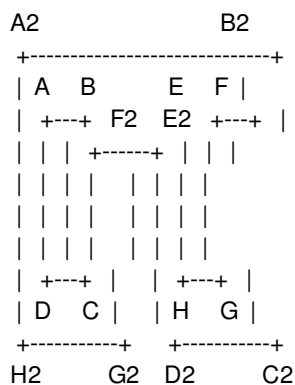
The following example oversizes the polygon A-B-C-D-E-F-G-H-A polygon by 10 units, producing the new polygon B1-C1-D1-E1-F1-G1-H1-A1-B1.

```
A1          B1
+-----+
```



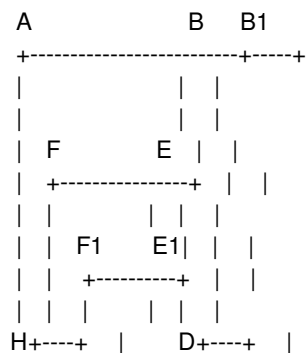
```
prompt> resize_polygon {{10 40} {60 40} {60 10} {50 10} {50 30} \
{20 30} {20 10} {10 10} {10 40}} -size 10
{{70.000 50.000} {70.000 0.000} {40.000 0.000} {40.000 20.000} {30.000 20.000}
{30.000 0.000} {0.000 0.000} {0.000 50.000} {70.000 50.000}}
```

The following example undersizes the polygon A2-B2-C2-D2-E2-F2-G2-H2-A2 polygon by 10 units, producing two polygons, A-B-C-D-A and E-F-G-H-E.



```
prompt> resize_polygon {{0 45} {80 45} {80 0} {50 0} {50 30} \
{30 30} {30 0} {0 0} {0 45}} -size -10
{{10.000 35.000} {20.000 35.000} {20.000 10.000} {10.000 10.000} {10.000 35.000}}
{{60.000 35.000} {70.000 35.000} {70.000 10.000} {60.000 10.000} {60.000 35.000}}
```

The following example oversizes the polygon A-B-C-D-E-F-G-H-A polygon on the right by 10 units and on the bottom by 10 units, producing the new polygon B1-A-H1-G1-F1-E1-D1-C1-B1




```

| G |   | C |
|   |   |   |
+-----+ +-----+
H1      G1      D1      C1

```

```

prompt> resize_polygon {{10 40} {60 40} {60 10} {50 10} {50 30} \
{20 30} {20 10} {10 10} {10 40}} -size_right 10 -size_bottom 10
{{70.000 40.000} {10.000 40.000} {10.000 0.000} {30.000 0.000} {30.000 20.000}
{50.000 20.000} {50.000 0.000} {70.000 0.000} {70.000 40.000}}

```

The following example passes the result of the **resize_polygon** command to another **resize_polygon** command. Note that undersizing by 10 and then oversizing by 10 does not restore the original polygon because of gap filling performed during undersizing.

```

prompt> resize_polygon -size -10 \
[resize_polygon {{10 40} {50 40} {50 10} \
{40 10} {40 30} {20 30} {20 10} {10 10} {10 40}} -size 10]
{{10.000 40.000} {50.000 40.000} {50.000 10.000} {10.000 10.000} {10.000 40.000}}

```

SEE ALSO

[convert_from_polygon\(2\)](#)
[compute_polygons\(2\)](#)

rewire_clock_gating

Changes the clock-gating cell implemented by the tool for a particular gated cell.

SYNTAX

```
status rewire_clock_gating  
  [-gating_cell new_clock_gating_cell]  
  [-gated_objects gated_objects_list]  
  [-balance_fanout]  
  [-undo]  
  [-verbose]
```

Data Types

```
new_clock_gating_cell  string  
gated_objects_list    list
```

ARGUMENTS

-gating_cell *new_clock_gating_cell*

Specifies a new clock-gating cell to be used to gate the cells or pins on the *gated_objects_list*.

-gated_objects *gated_objects_list*

Specifies that all registers, clock gates, and clock pins of gated modules in the *gated_objects_list* are gated by the *new_clock_gating_cell*.

-balance_fanout

Specifies that the clock gates and registers must be rewired to meet the minimum and maximum fanout constraints specified with the **set_clock_gating_style** command.

-undo

Indicates that directives specified by previously-executed **rewire_clock_gating** commands are to be removed.

-verbose

Specifies that additional information is to be displayed as the command executes.

DESCRIPTION

This command changes the clock-gating cell for the specified gated cell. The cell can be a register, a clock gate, or a gated module.

The tool performs clock gating at the RTL level with the use of the **compile_ultra -gate_clock** command. If you use the **set_clock_gating_style** command with the **-max_fanout** option, the tool limits the fanout of individual clock-gating elements created by **compile_ultra -gate_clock**. The tool duplicates clock-gating logic to ensure that the fanout of each element is bound by the constraint specified with the **-max_fanout** option. The clock-gating cells created to satisfy the **-max_fanout** constraint are logically equivalent. The **rewire_clock_gating** command directs the tool to move a clock-gated cell from one clock-gating cell to another logically-equivalent clock-gating cell.

The **rewire_clock_gating** command directs the tool to perform the actual circuit modifications to rewire clock gating in the design. You can undo the effect of any prior **rewire_clock_gating** command by using the **-undo** option. The **-undo** option deletes the directives previously specified by the **rewire_clock_gating** command.

In some cases, the tool could delete or add registers. This might alter the balanced fanout of clock gates. When invoked with the **-balance_fanout** option, **rewire_clock_gating** attempts to reinstate the clock gate fanout limits by splitting, merging or rewiring.

If multibit registers have to be rewired, fanout is computed by adding the total bit-width of each multibit register. For instance, if a clock gate is gating a 4-bit register and a 2-bit register, the total fanout of this clock gate is NOT 2, but 6.

When using this option, running **compile -incremental** might not be necessary. Under some circumstances, a few GTECH cells might be introduced. In this case, the **rewire_clock_gating** command prints a message that running **compile -incremental** is necessary. When the command is going to be invoked from a script (non-interactive mode) it is best to add **compile -incremental** after **rewire_clock_gating -balance_fanout**. This option can also be used if you require the clock-gate fanout limits to be changed after the **compile_ultra -gate_clock** command. Invoke this command after the **compile** command. Note that this command has no effect on multilevel and module clock gates.

Because rewiring gated registers alters the connections of the clock-gating cell that gates the registers, any path-based timing exception that passes through the old clock-gating cell to a gated register is no longer relevant and is lost.

If either the *new_clock_gating_cell* or any of the *gated_register_list* is marked for removal of clock gating with the **remove_clock_gating** command, the rewiring directive is not honored.

This command has no effect on a clock-gating cell or its gated registers if the clock gate or its parent hierarchical cell is marked as **dont_touch** with the **set_dont_touch** command.

EXAMPLES

The following example provides a directive to change the clock-gating cell gating the *pipeline_reg_A[0]* and *pipeline_reg_A[3]* registers to the *clk_gate_pipeline_reg_A_0* clock-gating cell:

```
prompt> rewire_clock_gating -gating_cell \
      clk_gate_pipeline_reg_A_0 -gated_registers \
      {pipeline_reg_A[0], pipeline_reg_A[3]}
```

The following example provides a directive to remove the previous directive and to change the clock-gating cell gating the *pipeline_reg_A[0]* and *pipeline_reg_A[3]* registers:

```
prompt> rewire_clock_gating -undo \
      -gated_registers {pipeline_reg_A[0] pipeline_reg_A[3] }
```

The following example directs the **rewire_clock_gating** command to change the minimum and maximum fanout limits to new values:

```
prompt> set_clock_gating_style -min 2 -max 128

prompt> rewire_clock_gating -balance_fanout
```

SEE ALSO

compile(2)
compile_ultra(2)
remove_clock_gating(2)
report_clock_gating(2)
set_false_path(2)
set_max_delay(2)
set_min_delay(2)
set_multicycle_path(2)

rp_group_inclusions

Returns a collection of relative placement groups that are directly included in the specified relative placement groups.

SYNTAX

collection **rp_group_inclusions**
[*rp_groups*]

Data Types

rp_groups list or collection

ARGUMENTS

rp_groups

Specifies the relative placement groups in which to search for directly-included relative placement groups.

If you do not specify this argument, the tool searches all relative placement groups for directly-included relative placement groups.

DESCRIPTION

The **rp_group_inclusions** command returns a collection containing all relative placement groups that are directly included in the relative placement groups specified in *rp_groups*. If you do not specify *rp_groups*, the command returns a collection containing all included relative placement groups. This command is supported only for designs that do not contain multiply-instantiated designs.

A group directly includes another group if the group was added by using the **add_to_rp_group-hierarchy** command without also using the **-instance** option.

If no groups are found, an empty string is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **rp_group_inclusions** command:

```
prompt> get_rp_groups  
{b::top a0::a0_group b::u_top/a1_group  
b::u_top/a1_group_include b::u_nxt/a1_group  
b::a1_group_include a1::a1_group}
```

```
prompt> rp_group_inclusions  
{b::u_top/a1_group b::u_nxt/a1_group}
```

```
prompt> rp_group_inclusions a0::a0_group
```

```
prompt> rp_group_inclusions top  
{b::u_top/a1_group b::u_nxt/a1_group}
```

```
prompt> remove_rp_groups -all -quiet  
1
```

```
prompt> rp_group_inclusions
```

SEE ALSO

- add_to_rp_group(2)
- all_rp_groups(2)
- all_rp_hierarchicals(2)
- all_rp_inclusions(2)
- all_rp_instantiations(2)
- all_rp_references(2)
- create_rp_group(2)
- remove_rp_groups(2)
- rp_group_instantiations(2)
- rp_group_references(2)

rp_group_instantiations

Returns a collection of relative placement groups that are instantiated in any of the specified relative placement groups.

SYNTAX

```
collection rp_group_instantiations  
[rp_groups]
```

Data Types

rp_groups list or collection

ARGUMENTS

rp_groups

Specifies the relative placement groups in which to search for instantiated relative placement groups.

If you do not specify this argument, the tool searches all relative placement groups for instantiated relative placement groups.

DESCRIPTION

The **rp_group_instantiations** command returns a collection containing all relative placement groups that are instantiated in the relative placement groups specified in *rp_groups*. If you do not specify *rp_groups*, the command returns a collection containing all instantiated relative placement groups.

This command is supported only for designs that do not contain multiply-instantiated designs.

A group instantiates another group if the group was added by using the **add_to_rp_group -hierarchy -instance** command.

If no groups are found, an empty string is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **rp_group_instantiations** command:

```
prompt> get_rp_groups
{b::top a0::a0_group b::u_top/a1_group
 b::u_top/a1_group_include b::u_nxt/a1_group
 b::a1_group_include a1::a1_group}

prompt> rp_group_instantiations
{a0::a0_group}

prompt> rp_group_instantiations b::a1_group_include
{a1::a1_group}

prompt> remove_rp_groups -all -quiet
1

prompt> rp_group_instantiations
```

SEE ALSO

- add_to_rp_group(2)
- all_rp_groups(2)
- all_rp_hierarchicals(2)
- all_rp_inclusions(2)
- all_rp_instantiations(2)
- all_rp_references(2)
- create_rp_group(2)
- remove_rp_groups(2)
- rp_group_inclusions(2)
- rp_group_references(2)

rp_group_references

Returns a collection of cells that are directly included in the specified relative placement groups.

SYNTAX

```
collection rp_group_references  
  [rp_groups]  
  [-leaf | -instance | -all_leaf]
```

Data Types

rp_groups list or collection

ARGUMENTS

rp_groups

Specifies the relative placement groups in which to find directly-included cells.

-leaf | -instance | -all_leaf

Specifies to return only leaf cells, only hierarchical cells, or all leaf cells in the group and down the hierarchy. By default, both leaf and hierarchical cells are returned.

DESCRIPTION

The **rp_group_references** command returns a collection containing all cells that are directly included in the relative placement groups specified in *rp_groups*. If you do not specify *rp_groups*, the command returns a collection containing all cells that are directly included in any relative placement group.

This command is supported only for designs that do not contain multiply-instantiated designs.

When you specify the **-leaf** option, only leaf cells that are contained in the specified relative placement groups are returned. When you specify the **-instance** option, only hierarchical cells are returned. When you specify the **-all_leaf** option, all cells that are in the specified relative placement groups and their hierarchical relative placement groups are returned.

An included relative placement group, unlike an instantiated relative placement group, does not have a hierarchical cell associated with it, so the **rp_group_references** command does not report included hierarchical relative placement groups.

A group directly includes a leaf cell if the cell was added to that group using the **add_to_rp_group -leaf** command.

A group directly includes a hierarchical cell if the cell was used to hierarchically instantiate another group using the

add_to_rp_group -instance command.

If no cells are found, an empty string is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **rp_group_references** command:

```
prompt> get_rp_groups
{b::top a0::a0_group b::u_top/a1_group
 b::u_top/a1_group_include b::u_nxt/a1_group
 b::a1_group_include a1::a1_group}
```

```
prompt> rp_group_references
{u_mid u_bot}
```

```
prompt> all_rp_references u_0 -design a0
{a0::a0_group}
```

```
prompt> rp_group_references a0::a0_group
{u_bot/u_1 u_bot/u_0}
```

```
prompt> remove_rp_groups -all -quiet
1
```

```
prompt> rp_group_references
```

The following examples show the difference between the **-leaf** and **-all_leaf** options. The top relative placement group **D::top** directly includes leaf cells *U1*, *U2*, *U3*, *U4*, and relative placement group **D::mid**. The **D::mid** relative placement group directly includes leaf cells *U11*, *U12*, *U15*, and *U16*.

```
prompt> rp_group_references -leaf D::top
{ U1, U2, U3, U4}
```

```
prompt> rp_group_references -all_leaf D::top
{ U1, U2, U3, U4, U11, U12, U15, U16}
```

SEE ALSO

add_to_rp_group(2)
all_rp_groups(2)
all_rp_hierarchicals(2)
all_rp_inclusions(2)
all_rp_instantiations(2)
all_rp_references(2)
create_rp_group(2)

```
remove_rp_groups(2)  
rp_group_inclusions(2)  
rp_group_instantiations(2)
```

run_test_point_analysis

Runs SpyGlass DFT testability analysis.

SYNTAX

status **run_test_point_analysis**

ARGUMENTS

This command has no arguments.

DESCRIPTION

The **run_test_point_analysis** command performs SpyGlass DFT analysis to determine the test points to be inserted by the **testability** DFT client.

Testability analysis is configured by the **set_testability_configuration** command. For details, see that man page.

This command is run after pre-DFT DRC, but before DFT preview and insertion.

Before running this command, the SPYGLASS_HOME environmental variable (not Tcl application variable) must be set so the tool can find the **spyglass** binary. You can check this as follows:

```
prompt> echo $env(SPYGLASS_HOME)
/global/apps/spyglass_2016.06-SP2
```

When you run this command, it reports information about the SpyGlass DFT analysis as it runs, such as coverage values and number of test points found.

The analysis is performed inside a SpyGlass DFT project directory whose name is uniquified using the current process ID and username. This directory name is reported when analysis starts:

```
Information: Test point analysis directory is '/proj/chip/_snpDft_user3.25145.0'.
```

The following testability targets do not require you to run testability analysis (although doing so is harmless):

- The **core_wrapper** target
- The **user** target
- Any target that has the **-only_from_file true** option set

EXAMPLES

The following example configures testability analysis, then reports the resulting test points:

```
# enable and configure testability analysis
set_dft_configuration -testability enable
set_testability_configuration -target {random_resistant}
set_testability_configuration -target {untestable_logic}

# perform pre-DFT DRC
create_test_protocol
dft_drc

# perform testability analysis
run_test_point_analysis

# preview the resulting test points
preview_dft -test_points all
```

SEE ALSO

```
preview_dft(2)
set_dft_configuration(2)
set_testability_configuration(2)
```

saif_map

Manages the SAIF name-mapping mechanism for reading SAIF files.

SYNTAX

```
string saif_map
  [-start]
  [-end]
  [-reset]
  [-report]
  [-get_name]
  [-set_name names]
  [-add_name names]
  [-remove_name names]
  [-clear_name]
  [-get_object_names name]
  [-create_map]
  [-write_map filename]
  [-read_map filename]
  [-type type]
  [-inverted]
  [-instances objects]
  [-no_hierarchical]
  [-columns columns]
  [-sort columns]
  [-rtl_summary]
  [-missing_rtl]
  [-input SAIF_file]
  [-source_instance SAIF_instance_name]
  [-target_instance target_instance_name]
  [-review]
  [-preview]
  [-hsep character]
  [object_list]
  [-nosplit]
```

Data Types

<i>names</i>	string
<i>name</i>	string
<i>filename</i>	string
<i>objects</i>	list
<i>columns</i>	string
<i>SAIF_file</i>	string
<i>SAIF_instance_name</i>	string
<i>target_instance_name</i>	cell
<i>character</i>	string

object_list list

ARGUMENTS

-start

Initializes the name-mapping database. The features of the name-mapping mechanism are not available unless it is initialized. Use the **saif_map -start** command before reading the RTL source files when you need to use the SAIF name-mapping mechanism.

-end

Reverses the initialization of the name-mapping database.

-reset

Resets the name-mapping information by clearing all SAIF names from the name-mapping database and sets all the original names to an initial value depending on the object's name and scope.

-report

Outputs a report of the name-mapping information. By default, the report lists the design objects and their SAIF names. You can use the **-columns** option to include more information, such as the object's original name. You can use the **-sort** option to specify sorting criteria. By default, all objects in the current instance hierarchy are considered for the report. You can specify a list of objects explicitly as the *object_list* argument or implicitly by using the **-instances**, and **-no_hierarchical** options.

-get_name

Returns the name of the specified object in the name-mapping database. The name returned depends on the value of the optional **-type** option.

- If the **-type** option is **saif_names** (the default value), a list of SAIF names on that object is returned.
- If the **-type** option is **orig_name**, the original name is returned.
- If the **-type** option is **extended_orig_name**, the original name is returned in the extended syntax.

-set_name *names*

Sets the name of the specified object in the name-mapping database. The name to be set depends on the value of the optional **-type** option.

- If the **-type** option is **saif_names** (the default value), the SAIF names on that object are set.
- If the **-type** option is **orig_name**, the original name is set.
- If the **-type** option is **extended_orig_name**, the original name is set and the *names* argument is expected to be in extended syntax.

-add_name *names*

Adds the specified SAIF names to the specified object in the name-mapping database.

-remove_name *names*

Removes the specified SAIF names from the specified object in the name-mapping database.

-clear_name

Clears the names of the specified object in the name-mapping database. The name that is cleared depends on the value of the optional **-type** option.

- If the **-type** option is **saif_names** (the default value), the SAIF names on that object are cleared.
- If the **-type** option is **orig_name** or **extended_orig_name**, the original name is cleared.
- If the **-type** option is **all_names**, both, the SAIF names and the original name are cleared.

You can specify the objects explicitly as the *object_list* argument or implicitly by using the optional **-instances**, and **no_hierarchical** options. If you do not specify the **-instances** option, the current instance is used.

-get_object_names *name*

Returns the object names of the design objects that have the specified name in the name-mapping database. The name used for finding the design objects in the name-mapping database depends on the value of the **-type** option.

- If the **-type** option is **saif_names** (the default value), the *name* argument is assumed to be a SAIF name.
- If the value of the **-type** option is **orig_name**, the *name* argument is assumed to be an original name.
- If the value of the **-type** option is **extended_orig_name**, the *name* argument is assumed to be an original name in extended syntax.

-create_map

Updates the name-mapping database automatically with new SAIF names by reading a SAIF file and matching design objects with the names in the SAIF file. The SAIF file must be specified by using the **-input** option, and the instance name of the current design in the SAIF file must be specified by using the **-source_instance** option.

-write_map *filename*

Outputs the name-mapping database into the specified name-mapping file. You can read the name-mapping file back into Power Compiler by using the **-read_map** option of the **saif_map** command. If the **-type primpower** option or **-type ptpx** option is used, the SAIF names in the name-mapping database are output into a file with a list of **set_rtl_to_gate_name** commands that can be read by PrimePower or PrimeTime PX.

-read_map *filename*

Reads the name-mapping information stored in a name-mapping file generated by using the **-write_map** option of **saif_map**.

-type *type*

Specifies an optional type argument with some of the **saif_map** command features. When used with the **-get_name**, **-set_name**, or **-get_object_names** options, the following values are the valid type keywords:

- **saif_names** (the default) specifies that SAIF names are used.
- **orig_name** specifies that original names are used.
- **extended_orig_name** specifies that original names in extended syntax format are used.

In addition to these values, you can use the **all_names** keyword with the **-clear_name** option to specify that both SAIF names and original names are cleared.

When used with the **-write_map** option, the following values are the valid type keywords:

- **name_map** (the default) specifies that a name-mapping file is created.
- **ptpx** specifies that a set of PrimeTime PX instructions is generated.
- **primpower** specifies that a set of PrimePower instructions is generated.

-inverted

Specifies that the name to be set, maps to a logically inverted object. This option is valid only with the **-set_name** option. For example, you can use the **-inverted** option when specifying that the object *A_reg/QN* maps to the SAIF named *A*, but its logical value is inverted. However, note that for such a simple example, it is generally sufficient to specify that the register cell *A_reg* maps to the SAIF named *A*, and the fact that *A_reg/QN* is logically inverted to *A* is deduced automatically when the SAIF file is read.

-instances *objects*

Restricts the effect of some of the **saif_map** command features to the specified instances. Some **saif_map** options (including **-report** and **-clear**) act on a number of objects that can be specified explicitly as the *object_list* argument or implicitly by using the optional **-instances**, or **-no_hierarchical** options. The *objects* argument is a list of hierarchical cells, and the default value is the current instance. If you do not explicitly specify the list of objects, all the objects in the current instance are used (if you use **-no_hierarchical**, only the top-level objects are used).

-no_hierarchical

Specifies that all objects in the top-level of the current instance or the instances specified with the **-instance** option are used by the **saif_map** command. Some **saif_map** options (including **-report** and **-clear**) act on a number of objects that can be specified explicitly as the *object_list* argument or implicitly by using the optional **-instances**, or **-no_hierarchical** options.

-columns *columns*

Specifies a list of column names that are used when reporting the name-mapping database information. The possible values of the elements in the *columns* argument are

- **type** specifies the type (pin, port, cell, or net) of the object.
- **object_names** specifies the name of the design object.
- **saif_names** specifies that SAIF names are included in the report.
- **orig_names** specifies that original names are included in the report.
- **extended_orig_names** specifies that original names in extended format are included in the report.
- **attributes** specifies that attributes, or flags, are included in the report.

The default value of *columns* is {type object_names saif_names attributes}.

-sort *columns*

Specifies the sorting criteria that is used when reporting the name-mapping database information. The *columns* argument is a list of column names as specified under the **-columns** description above.

-rtl_summary

Specifies that a summary report of which RTL or synthesis invariant objects have SAIF name-mapping information. This option is valid only when you also use the **-report** option. The report contains the number of synthesis invariant design objects that have SAIF name information obtained automatically or set by you. It also contains the number of design objects that do not have the SAIF name information annotated directly on them, but annotated to some of the object's connecting nets, pins, or ports. The synthesis invariant objects included in the report are design ports, hierarchical cell pins, sequential cells, and tristate cells.

-missing_rtl

Displays a list of RTL or synthesis invariant objects that do not have SAIF name-mapping information. This option is valid only when you also use the **-report** and **-rtl_summary** options.

-input *SAIF_file*

Specifies the name of the SAIF file that is used to create the name-mapping information when used with the **-create_map** option.

-source_instance *SAIF_instance_name*

Specifies the name of the current design instance in the SAIF file that is used to create the name-mapping information when used with the **-create_map** option.

-target_instance *target_instance_name*

Specifies the instance in the current design for which the name-mapping information is created when used with the **-create_map** option. The default target instance is the current instance.

-review

Displays a SAIF match report showing which design objects are matched with which SAIF file objects after the SAIF name-mapping information is created from a SAIF file. This option is valid only with the **-create_map** option.

-preview

Displays a SAIF match report without creating the actual SAIF name mapping. The report shows which design objects would be matched with which SAIF file objects when the name mapping is created. This option is valid only when used with the **-create_map** option.

-hsep *character*

Specifies the hierarchical separator when parsing or reporting SAIF names and original names. The default value is slash (/). Other possible values are dot (.) and double quotation marks (" "). The **-hsep " "** argument can be useful when the slash (/) character is part of the identifier names of the design objects.

object_list

Specifies the list of objects used by the **saif_map** command.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column-width, the next field begins on a new line, starting in the correct column.

OBSOLETE ARGUMENTS

-hierarchical

When selecting objects implicitly, the **-hierarchical** flag is assumed by default, so it is never required.

DESCRIPTION

The **saif_map** command manages and uses a SAIF name-mapping database that provides a mapping between design objects and the names in SAIF files. The name-mapping database can be created automatically and you can query and modify the information in the database. You can use the name-mapping database when reading SAIF files generated by RTL simulation, where the names in the SAIF file do not exactly match the names of the design objects.

The information stored in the name-mapping database consists of the following possible information on design ports, nets, pins, and cells:

- A list of SAIF names, representing the possible names in the SAIF file that match the design object.
- An original name, representing the original name of the design object as created by the synthesis RTL reader.
- A number of flags that specify, for example, whether the SAIF and original names have been created automatically or set by the user.

When the name-mapping mechanism is active (the **saif_map -start** command activates the name-mapping mechanism), the original names are automatically created when the design RTL source is read and linked. The SAIF names and original names are full instance names.

The SAIF names can be created automatically when processing an RTL SAIF file by using the **saif_map -create_map** command, which matches the original names of the design objects with the names in the SAIF file. The SAIF names information can then be used when reading the SAIF file by using the **-map_names** option of the **read_saif** command. Also, the name-mapping mechanism can be created automatically and used to read a SAIF file by using the **-auto_map_names** option of the **read_saif** command.

The **saif_map** command provides several options for reporting, querying, and modifying the SAIF name-mapping database information (see the list of arguments above). You might need to set the SAIF names of design objects explicitly by using the **-create_map** option in cases where they cannot be created automatically. You can report, query, and modify the original names of the design objects that can be represented in a simple format by using the **-type orig_name** option or in an extended syntax format by using the **-type extended_orig_name** option, which is close to the internal representation of the original names information and provides more information on the scope of the design objects.

You can write the name-mapping database information to a file by using the **-write_map** option. You can read the information back into the design by using the **-read_map** option. Note that the name-mapping database is stored on the design and is therefore automatically stored in the .ddc files. If you read the design as a gate-level source file, you might need to read the name-mapping information read back into the design.

EXAMPLES

The following example shows how you can automatically create and use the name-mapping mechanism when reading an RTL SAIF file into a compiled design.

```
saif_map -start
prompt> read_verilog ex.v
prompt> current_design ex
prompt> link
prompt> compile
prompt> read_saif -input ex.rtl.saif -instance tb/u1 -auto_map_names
```

The following example shows how you can create the name-mapping mechanism, report it, modify it slightly, and use it to read an RTL SAIF file.

```
prompt> saif_map -start
prompt> read_verilog ex.v
prompt> current_design ex
prompt> link
prompt> compile
prompt> saif_map -create_map -input ex.rtl.saif -source tb/u1
prompt> saif_map -report
prompt> saif_map -get_name [get_port clk_1]
prompt> saif_map -set_name "clk1" [get_port clk_1]
prompt> read_saif -input ex.rtl.saif -instance tb/u1 -map_names
```

The following example shows how the SAIF name-mapping information can be stored to a file and used later. Because the

saif_map -write command is used before the SAIF name mapping is created by using **saif_map -create_map**, there are no SAIF names in the database and only the original names are stored in the name-mapping file.

```
prompt> saif_map -start
prompt> read_verilog ex.v
prompt> current_design ex
prompt> link
prompt> compile
prompt> change_name -rules verilog
prompt> write -f verilog -hier -out ex.gate.v
prompt> saif_map -write_map ex.namemap

...

prompt> read_verilog ex.gate.v
prompt> saif_map -read_map ex.namemap
prompt> read_saif -input ex.rtl.saif -instance tb/u1 -auto_map_names
```

SEE ALSO

read_saif(2)
report_saif(2)
merge_saif(2)

save_qtm_model

Saves the current quick timing model (QTM) description.

SYNTAX

```
string save_qtm_model  
[-format db | lib | script]
```

ARGUMENTS

-format db | lib | script

Specifies the output format to be used.

Valid output formats are

- db - The .db version of the QTM model. By default, the .db format is used.
- lib - The .lib version of the QTM model.
- script - A script containing QTM commands. The command writes out a script file that contains the QTM commands to describe the QTM model.

DESCRIPTION

This command creates an in-memory .db timing model for the QTM model and indicates that the model being defined is completed. All QTM commands must be placed between a **create_qtm_model** command and a **save_qtm_model** command.

This command creates an in-memory Synopsys .db representation to be used by the tool's timing engine. This command runs faster than the obsolete **write_qtm_model** command and does not require disk I/O. You can call this command multiple times in a session if you need to modify the QTM model.

To display information about the current QTM model, use the **report_qtm_model** command.

For a basic description of QTM models, see the **create_qtm_model** man page. For a more detailed description about QTM models, see the *IC Compiler Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example saves the QTM model in .db and .lib format.

```
prompt> save_qtm_model -format {db lib}
```

The following example save the QTM model with a script of QTM commands.

```
prompt> save_qtm_model -format {script}
```

SEE ALSO

[create_qtm_model\(2\)](#)

[report_qtm_model\(2\)](#)

save_upf

Writes out the design's UPF power intent as a UPF command script.

SYNTAX

```
collection save_upf  
  [upf_file_name]  
  [-supplemental supf_file_name]  
  [-include_supply_exceptions]  
  [-full_upf]
```

Data Types

<i>upf_file_name</i>	string
<i>supf_file_name</i>	string

ARGUMENTS

upf_file_name

Specifies the name of the UPF script file written by the command.

This option can be used only in the UPF-prime mode, not the golden UPF mode.

-supplemental *supf_file_name*

Specifies the name of the supplemental UPF file written by the command.

This option can be used only in the golden UPF mode, not the UPF-prime mode.

-include_supply_exceptions

Specifies whether the supplemental UPF file written by the command contains exception power connections derived by the tool. Use this option when the Verilog netlist file does not contain any PG connection information.

This option can be used only in the golden UPF mode, not the UPF-prime mode.

-full_upf

In the hierarchical flow mode, **save_upf** excludes UPF commands that are defined in abstract hierarchies, or refer to elements in these hierarchies. This option specifies that **save_upf** should write all commands.

This option can be used only in the UPF-prime mode, not the golden UPF mode.

DESCRIPTION

In the UPF-prime (not golden) mode, the **save_upf** command writes out design's complete power intent infrastructure as a UPF command script. The script consists of commands previously loaded with the **load_upf** command, UPF commands entered at the tool prompt during the session, and UPF power intent changes derived by the tool during the session.

For example, if optimization has changed the name of the cell that drives an isolation signal, it modifies the **set_isolation_control** command to reflect the new isolation signal name. The script written by the **save_upf** command reflects the change made by the tool.

In the golden UPF mode, this command writes out a supplemental UPF file that reflects only the UPF changes made by the tool. The original "golden" UPF file and the supplemental UPF file together fully specify the UPF power intent of the design.

In the golden UPF mode, you can use the **-include_supply_exceptions** option to specify whether the supplemental UPF file contains supply connection information. You should use this option whenever you write the Verilog netlist without power supply connection information.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples saves the full UPF power intent of the design in a file called `my_design_power.upf`:

```
prompt> save_upf my_design_power.upf
```

In the following example, the tool is operating in the golden UPF mode. The command saves the supplemental UPF into a file called `my_supp_power.upf`.

```
prompt> save_upf -supplemental my_supp_power.upf
```

SEE ALSO

- `load_upf(2)`
- `check_upf(2)`
- `write_file(2)`
- `enable_golden_upf(3)`

select_block_scenario

Performs scenario mapping between the top-level design and interface logic models (ILMs) or block abstractions.

SYNTAX

```
status select_block_scenario  
  [-scenarios top_scenario_list]  
  [-block_references block_list]  
  [-block_scenario block_scenario | -reset
```

Data Types

```
top_scenario_list  list  
block_list         collection  
block_scenario    string
```

ARGUMENTS

-scenarios *top_scenario_list*

Specifies a list of top-level scenarios to which this mapping applies. The default is the current scenario. In a non-multicorner-multimode top-level design, you must not specify this option because the top-level design does not have any scenarios.

-block_references *block_list*

Specifies a list of block references for which the mapping is specified. The value can be a collection of block references or name patterns.

If this option is not specified, the command sets or resets the mapping for all the block references in the current design.

-block_scenario *block_scenario*

Specifies the scenario name in the specified blocks to which the top-level scenarios are mapped.

This option and the **-reset** option are mutually exclusive.

-reset

Resets the user-specified scenario mapping to name-based matching.

This option and the **-block_scenario** option are mutually exclusive.

DESCRIPTION

The **select_block_scenario** command enables you to select the ILM or block abstraction scenario from which the data is to be loaded.

Use this command to map the scenarios in the top-level design with the block-level scenarios in the following situations:

- When the names of the scenarios at the top-level and the block-level do not match
- To use a multicorner-multimode block in a non multicorner-multimode top-level design
- When you have additional modes in the top-level design than in the block

You do not need to use the **select_block_scenario** command with a non multicorner-multimode block and a multicorner-multimode top-level design. If the block is not multicorner-multimode, the tool automatically detects it and assumes that every top scenario is mapped to the block's operating conditions.

If you specify multiple top-level scenarios and multiple block references, the mapping applies to all specified top-level scenarios and for all specified blocks.

These settings are stored in the database when the top-level design is saved. The settings are restored when the design is later reopened. You need not specify these settings in every session.

You should set these mappings before applying constraints and avoid changing them later.

The tool expects the TLUPlus settings and temperature to be the same for the top-level scenario and the corresponding ILM or block abstraction scenario.

See the report generated by the **report_ilm** command or **report_block_abstraction** command to verify the scenario mappings that are specified.

You can reset the specified scenario mapping to name-based matching by using the **-reset** option.

Multicorner-Multimode Support

By default, this command uses information from the current scenario. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following example maps the top-level scenario top_scen to the block scenario blk_scen:

```
prompt> select_block_scenario -scenarios top_scen \
      -block_references blk -block_scenario blk_scen
1
```

The following example shows the command usage when the top-level design does not have scenarios and the block has a scenario named blk_scen. In this case, you must not specify the **-scenarios** option:

```
prompt> select_block_scenario \
      -block_references blk -block_scenario blk_scen
1
```

The following example resets the user-specified scenario mapping for the top-level scenario top_scen and blk block reference:

```
prompt> select_block_scenario -scenarios top_scen \
      -block_references blk -reset
1
```

SEE ALSO

`all_scenarios(2)`
`current_scenario(2)`
`report_block_abstraction(2)`

set_active_scenarios

Specifies which scenarios are to be active.

SYNTAX

```
status set_active_scenarios  
    scenario_list | -all
```

Data Types

scenario_list list of strings

ARGUMENTS

scenario_list

Specifies which scenarios are active.

This argument and the **-all** option are mutually exclusive. You must specify one, but not both.

-all

Makes all scenarios active.

This option and the *scenario_list* argument are mutually exclusive. You must specify one, but not both.

DESCRIPTION

This command specifies the scenarios that are to be active.

To report timing results with back-annotated delays for newly-activated scenarios, run the **extract_rc -estimate** command before issuing the **report_timing** command. Creation and usage of scenarios is available with Design Compiler Graphical.

Multicorner-Multimode Support

This command uses information from both active and inactive scenarios.

EXAMPLES

The following example uses **set_active_scenarios** to set only the *MODE1* scenario as active:

```
prompt> create_scenario MODE1
```

```
prompt> create_scenario MODE2
```

```
prompt> set_active_scenarios {MODE1}
```

```
prompt> all_scenarios  
MODE1 MODE2
```

```
prompt> all_active_scenarios  
MODE1
```

SEE ALSO

- `all_active_scenarios(2)`
- `all_scenarios(2)`
- `create_scenario(2)`
- `current_scenario(2)`
- `extract_rc(2)`
- `remove_scenario(2)`

set_ahfs_options

Specifies the options to use when running automatic high-fanout synthesis (AHFS).

SYNTAX

```
status set_ahfs_options  
  [-enable_port_punching true | false]  
  [-no_port_punching cells]  
  [-default_reference references]  
  [-port_map_file file_name]  
  [-preserve_boundary_phase true | false]  
  [-constant_nets true | false]  
  [-global_route true | false]  
  [-default]
```

Data Types

```
cells      string  
references string  
file_name  string
```

ARGUMENTS

-enable_port_punching true | false

Enables or disables port punching in automatic high-fanout synthesis optimization. The default is **true**.

When this option is **true**, automatic high-fanout synthesis can insert buffers or inverters across hierarchy boundaries to get better QoR.

Port punching is turned off in **compile_ultra** if the **-no_boundary_optimization** option is defined for **compile_ultra**.

You can use the **-no_port_punching** option to prevent port punching on specific hierarchical cells, even when **-enable_port_punching** is **true**.

-no_port_punching *cells*

Specifies the hierarchical cells for which automatic high-fanout synthesis cannot add or remove any ports. The cells are specified as a string of hierarchical cell instance names separated by commands, such as "sub/mod1 sub/block2".

Use this option to limit the port punching on specific hierarchical cell instances when **-enable_port_punching** is **true** (the default). If **-enable_port_punching** is **false**, there is no need for this option.

During the removal phase of automatic high-fanout synthesis, inverters might be moved across hierarchical boundaries to cancel "inverter chains" for better QoR. If you do not want to allow any phase changes on hierarchical boundaries, use the **-preserve_boundary_phase true** option instead of using this option.

-default_reference references

Specifies a list of buffers or a list of inverters for automatic high-fanout synthesis to use when constructing new buffer or inverter trees.

You can specify the reference cells with or without the library prefix.

If you do not specify this option, or if the specified references are not active buffer cells, automatic high-fanout synthesis tries to use all active buffer or inverter cells defined in the library.

This option is not supported by the **place_opt** command.

-port_map_file file_name

Specifies the name of the port-mapping file. This option is used only when **-enable_port_punching** is set to **true**.

If you do not specify this option and **-enable_port_punching** is set to **true**, the default name of the port-mapping file is *design_name_port_map.version*. The initial file is created with a version of 0. If the specified port-mapping file already exists, a new version of the file is created with the version number increased by one, such as *design_name_port_map.1*.

-preserve_boundary_phase true | false

Specifies whether automatic high-fanout synthesis is allowed to move inverters across hierarchical boundaries so that the logical phase of the boundary changes.

The default is **false**, meaning that automatic high-fanout synthesis can freely move inverters across hierarchical boundaries. This default behavior often leads to better QoR because automatic high-fanout synthesis can move inverters and cancel inverter chains as needed.

-constant_nets true | false

Specifies that automatic high-fanout synthesis works on constant nets as well as signal nets.

By default, automatic high-fanout synthesis ignores constant nets.

-global_route true | false

Specifies whether to enable Zroute-based global buffering during automatic high-fanout synthesis. This feature is deprecated and will be removed in a future release.

When this option is set to **true** before compile, Zroute-based global buffering takes effect during the compile and incremental compile stages. You must specify the **-spg** option with the **compile_ultra** command to enable it. For best results, if you enable Zroute-based global buffering in Design Compiler Graphical, enable it also in IC Compiler. Zroute-based global buffering might increase runtime and affect area and timing; therefore, use it only on macro-intensive designs with narrow channels.

-default

Resets all of the options set by previous **set_ahfs_options** commands to their default values.

DESCRIPTION

This command defines the constraints used when performing automatic high-fanout optimization. You can report the settings by running the **report_ahfs_options** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to set automatic high-fanout synthesis options for the current design. It enables port punching and uses cells of type BUFX3M or ss_0v80_125c/DLY4X4M during high-fanout synthesis. The **-preserve_boundary_phase false** option indicates that inverters should not move across hierarchical boundaries. The **-no_port_punching "sub/mod1 sub/mod2"** option indicates that the *sub/mod1* and *sub/mod2* hierarchical cells should not have changes to their ports during automatic high-fanout synthesis.

```
prompt> set_ahfs_options \  
-enable_port_punching true \  
-no_port_punching "sub/mod1 sub/mod2" \  
-preserve_boundary_phase false \  
-default_reference "BUFX3M ss_0v80_125c/DLY4X4M"
```

```
prompt> report_ahfs_options
```

Report AHFS options:

AHFS options for the design:

Enable port punching: ON

Default Reference : BUFX3M ss_0v80_125c/DLY4X4M

Port Map File :

Preserve Boundary Phase : OFF

No Port Punching on these hier cells : "sub/mod1 sub/mod2"

Constant Nets allowed during AHFS optimization : OFF

SEE ALSO

report_ahfs_options(2)

set_always_on_cell

Sets on-the-fly specification for an always-on library cell.

SYNTAX

```
status set_always_on_cell  
      cell_name
```

Data Types

```
cell_name  string
```

ARGUMENTS

cell_name

Specifies that the library cell is an always-on cell.

DESCRIPTION

The **set_always_on_cell** command sets all cells in the library whose name matches the specified cell name as always-on cells. Only buffer or inverter cells in the library can be specified as always-on cells.

EXAMPLES

The following example uses the **set_always_on_cell** command to set the library cell named *INV_AO* as an always-on cell:

```
prompt> set_always_on_cell INV_AO
```

SEE ALSO

```
check_library(2)  
report_lib(2)
```

set_always_on_strategy

Sets the always-on strategy for specified shutdown power domains.

SYNTAX

```
status set_always_on_strategy  
-object_list list_of_domains  
-cell_type single_power | dual_power
```

Data Types

list_of_domains collection

ARGUMENTS

-object_list *list_of_domains*

Specifies one or more power domains for which to define the always-on strategy.

-cell_type single_power | dual_power

Specifies the type of always-on cells to be used, either single-power cells or dual-power cells with backup power. The default is **dual_power**.

DESCRIPTION

This command sets the always-on strategy for specified shutdown power domains. The strategy indicates the type of cells to be used for always-on buffering. You can choose to use standard cells with single-power cells, or dual-power cells with a backup supply. This information drives cell selection during optimization.

The dual-power strategy uses special-purpose, always-on cells. Each of these cells has two power pins: a primary pin connected to the primary supply and a backup pin connected to a backup supply net. The cell also has an attribute called `always_on` which is set to true at the cell level. This type of cell is known as a dual-rail, always-on cell.

The single-power strategy uses standard, single-power cells that are placed in dedicated always-on site rows of the power domain's corresponding voltage area. These always-on sites are powered by the backup power supply. During optimization, the tool does not make any changes to these always-on site rows, so it leaves these cells on the always-on paths.

The dual-power strategy does not require always-on site rows and the cells can be placed anywhere in the voltage area.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following example sets the always-on strategy on *switchable_domain* to use single-power standard cells:

```
prompt> set_always_on_strategy \  
        -object_list switchable_domain \  
        -cell_type single_power  
1
```

SEE ALSO

`create_power_domain(2)`
`remove_power_domain(2)`
`report_power_domain(2)`
`set_power_guide(2)`
`unset_power_guide(2)`

set_analyze_rtl_logic_level_threshold

Specifies the logic level threshold value to be used by report_logic_levels command. Also used to remove the previously set value.

SYNTAX

```
status set_analyze_rtl_logic_level_threshold
  [-group group_name]
  [-reset]
  [threshold]
```

Data Types

```
group_name  string
threshold   integer
```

ARGUMENTS

-group *group_name*

Specifies a name for the path group for which threshold value is being set or reset. The group name should be one of the group names reported by report_path_group command. When this option is not used, the threshold setting is applied at global level.

-reset

Specifies that the previously specified threshold value be reset. If -group option is specified, then reset applies only to the specified path group. If -group option is not specified, then the threshold value is reset for all path groups and global level.

threshold

Specifies the threshold integer value to be used by report_logic_levels command.

DESCRIPTION

This command is used to specify the threshold value to be used by report_logic_levels command. The value can be specified at each path group level or at global level. When threshold is specified both at global level and path group level, path group threshold setting is used for the paths belonging to that path group.

Not using this command will result in tool computing the threshold automatically based on the required time for the path and the delay of average size NAND2 cell in the target library.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the logic level threshold to 10 at global level.

```
prompt> set_analyze_rtl_logic_level_threshold 10
```

The following example sets the logic level threshold to 5 only for the paths in the path group CLOCK while setting 6 for all other paths.

```
prompt> set_analyze_rtl_logic_level_threshold 6  
prompt> set_analyze_rtl_logic_level_threshold -group CLOCK 5
```

The following example re-sets the logic level threshold for path group CLOCK.

```
prompt> set_analyze_rtl_logic_level_threshold -group CLOCK -reset
```

The following example re-sets the logic level threshold for all paths (all path groups).

```
prompt> set_analyze_rtl_logic_level_threshold -reset
```

SEE ALSO

report_logic_levels(2)
report_path_group(2)

set_annotated_check

Sets the setup, hold, recovery, or removal timing check value between two pins.

SYNTAX

```
status set_annotated_check
    check_value
    -from from_pins
    -to to_pins
    -setup
    | -hold
    | -recovery
    | -removal
    | -nochange_high
    | -nochange_low
    [-rise | -fall]
    [-clock rise | fall]
    [-worst]
    [-increment]
```

Data Types

```
check_value    float
from_pins      list
to_pins        list
```

ARGUMENTS

check_value

Specifies the timing check value between pins on the same cell, which can be positive or negative. You must express the *check_value* argument in units consistent with the technology library used during optimization. For example, if the technology library specifies delay values in nanoseconds, the *check_value* argument must be expressed in nanoseconds.

-from *from_pins*

Specifies the leaf-cell clock pins that are the startpoints of the timing arcs for which checks are to be annotated.

-to *to_pins*

Specifies the leaf-cell data pins that are the endpoints of the timing arcs for which checks are to be annotated.

-setup | -hold | -recovery | -removal | -nochange_high | -nochange_low

Specifies the type of the timing check. There must be a corresponding timing arc between the from and to pins.

- The **-setup** option specifies that data must be stable for the amount of time specified by *check_value* before the closing clock edge.
- The **-hold** option specifies that data must be stable for the amount of time specified by *check_value* after the closing clock edge.
- The **-recovery** option specifies that an asynchronous set or clear inactive edge cannot occur within the amount of time specified by *check_value* before the closing clock edge. This is essentially a setup requirement on an asynchronous pin, but only the inactive transition is considered.
- The **-removal** option specifies that an asynchronous set or clear inactive edge cannot occur until the amount of time specified by *check_value* after the closing clock edge. This is essentially a hold requirement on an asynchronous pin, but only the inactive transition is considered.
- The **-nochange_high** option provides a timing check that specifies that the data must not rise within the amount of time specified by *check_value* before the clock becomes active and must not fall until the amount of time specified by *check_value* after the clock becomes inactive.
- The **-nochange_low** option provides a timing check that specifies that the data must not fall within the amount of time specified by *check_value* before the clock becomes active and must not rise until the amount of time specified by *check_value* after the clock becomes inactive.

-rise | -fall

Specifies whether the check is for the data rise or fall transition. If you do not specify either **-rise** or **-fall**, both values are set.

The **-rise** option specifies a rise data transition that corresponds to the check before the activating clock edge. A rise data transition corresponds to the check after the deactivating clock edge. A rise clock transition indicates that the clock is active-high.

The **-fall** option specifies a fall data transition that corresponds to the check after the deactivating clock edge. A fall data transition corresponds to the check before the activating clock edge. A fall clock transition indicates that the clock is active-low.

-clock rise | fall

Specifies whether the check is for clock rising or falling. By default, checks for both clock rise and fall are set.

-worst

Specifies that the check is to overwrite the previously annotated check only if the *check_value* specified is worse than the check value previously annotated. By default, *check_value* overwrites any previously annotated value.

-increment

Adds the specified check value to the existing default check value. Without this option, the check value replaces the default check value. If you use the **set_annotated_check** command multiple times with the **-increment** option, the last **-increment** setting is used and earlier ones are ignored. If you use a **set_annotated_check** command without the **-increment** option, that check value is used, and any **-increment** setting is ignored.

DESCRIPTION

This command annotates a timing check between two or more pins on a cell or a net in the current design. This might be appropriate after place and route for technologies where the timing check value varies for different instances and the library timing check values do not provide sufficient accuracy.

If the design is not already linked, the **set_annotated_check** command links it automatically.

The command can be used for pins at lower levels of the design hierarchy. Pins are specified as

"INSTANCE1/INSTANCE2/PIN_NAME."

To list annotated timing check values, use the **report_annotated_check** command. To remove annotated timing check values from a design, use the **remove_annotated_check** or **reset_design** command. To see the affect of **set_annotated_check** for a specific instance, use the **report_timing** command.

Multicorner-Multimode Support

This command applies to current scenario only.

EXAMPLES

The following example annotates a setup time of 2.1 units between the CP clock pin and the D data pin of the u1/ff12 cell instance:

```
prompt> set_annotated_check -setup 2.1 \  
-from u1/ff12/CP -to u1/ff12/D
```

The following example annotates a nochange check between data low and clock low. The nochange margin before the clock is 1.3 units and the nochange margin after the clock is 2.4 units.

```
prompt> set_annotated_check -nochange_low 1.3 \  
-clock fall -fall -from u1/CP -to u1/EN
```

```
prompt> set_annotated_check -nochange_low 2.4 \  
-clock fall -rise -from u1/CP -to u1/EN
```

SEE ALSO

current_design(2)
link(2)
remove_annotated_check(2)
report_annotated_check(2)
report_timing(2)
reset_design(2)

set_annotated_delay

Sets the net or cell delay value between two pins.

SYNTAX

```
status set_annotated_delay
-net | -cell
[-dont_touch]
[-load_delay load_delay_type]
[-rise | -fall]
[-min]
[-max]
delay_value
[-from from_pins]
[-to to_pins]
[-worst]
```

Data Types

```
load_delay_type  string
delay_value     float
from_pins        list
to_pins          list
```

ARGUMENTS

-net

Specifies a net delay, between a driver and a receiver.

-cell

Specifies a cell delay, from an input to an output of a cell.

You must specify either **-net** or **-cell**, and not both.

-dont_touch

Sets the **dont_touch** attribute to **true** for the specified net. By default, this attribute is set to **false**, allowing optimization to change the net and remove the annotated delay constraint. If you use the **-dont_touch** option, the net cannot be changed by optimization. This option works only with the **-net** option, not with the **-cell** option. You can report the **dont_touch** attribute setting with the **report_dont_touch -class net** command.

-load_delay *load_delay_type*

Specifies whether to include load delay as part of annotated net delays or as part of annotated cell delays. The *load_delay_type*

must be either **net** or **cell**. Load delay is the portion of cell delay arising from the capacitive load of the net the cell is driving. All timing arcs of the same net or same cell must be annotated with the same *load_delay_type*.

-rise

Applies the delay to rising transitions.

-fall

Applies the delay to falling transitions.

If you specify neither **-rise** nor **-fall**, the delay value applies to both rising and falling transitions.

-min

Specifies that the *delay_value* is to be used for minimum delay analysis.

-max

Specifies that the *delay_value* is to be used for maximum delay analysis.

By default, the *delay_value* is used for both maximum and minimum delay analysis.

delay_value

Specifies the delay value between pins on the same cell or net. The *delay_value* must be expressed in units consistent with the logic library used during optimization. For example, if the logic library specifies delay values in nanoseconds, *delay_value* must also be in nanoseconds.

-from *from_pins*

Specifies a list of leaf-cell pins or top-level ports that are the startpoints of the timing arcs for which delays are to be annotated.

If you omit the **-from** option, all arcs to the *to_pins* are annotated.

-to *to_pins*

Specifies a list of leaf-cell pins or top-level ports that are the endpoints of the timing arcs for which delays are to be annotated.

If you omit the **-to** option, all arcs from the *from_pins* are annotated.

-worst

Overwrites the previously annotated delay only if the *delay_value* specified is worse than the delay value previously annotated. By default, the *delay_value* overwrites any previously annotated value.

DESCRIPTION

This command annotates cell or net delays in the design.

Cell delays exist between pins of the same leaf cell. With the **-cell** option, pins in *from_pins* must be cell input or inout pins, and pins in *to_pins* must be cell output or inout pins.

Net delays exist between leaf-cell pins or top-level ports connected by a net. With the **-net** option, pins in *from_pins* must be cell output or inout pins or top-level input ports, and pins in *to_pins* must be cell input or inout pins or top-level output ports.

To verify the accuracy of the back-annotation done by the **set_annotated_delay** command, run the **update_timing** command.

Delays annotated with this command affect both timing analysis and optimization.

Load delay, also known as extra source gate delay, is the portion of cell delay caused by the capacitive load of the net being driven. Some delay calculators consider load delay part of the net delay and others consider it part of the cell delay. If your annotated delay value (for either cell or net) assumes that load delay is part of the cell delay, use **-load_delay cell**. If your delay value assumes that load delay is included in the net delay, use **-load_delay net**. By default, load delays are assumed to be in cell delays, so they appear in **report_timing** path listings.

The specified delay value overrides the internally estimated cell and net delay value. If the specified pins are not in the same cell or on the same net, when the timing is updated, the tool issues an error message and *delay_value* is discarded for those pins.

You can use **set_annotated_delay** for pins at lower levels of the design hierarchy by specifying them as "INSTANCE1/INSTANCE2/PIN_NAME."

To list annotated delay values, use the **report_annotated_delay** command. To remove the annotated cell or net delay values from a design, use the **remove_annotated_delay** or **reset_design** command.

Multicorner-Multimode Support

This command applies to current scenario only.

EXAMPLES

The following example annotates a cell delay of 20.0 time units between input pin A of cell instance U1/U2/U3 and output pin Z of the same cell instance. The delay value of 20.0 includes the load delay.

```
prompt> set_annotated_delay -cell -load_delay cell 20.0 \
      -from U1/U2/U3/A -to U1/U2/U3/Z
```

The following example annotates a rise net delay of 1.4 between output pin U1/Z and input pin U2/A. The delay value for this net does not include load delay.

```
prompt> set_annotated_delay -net -rise -load_delay cell 1.4 \
      -from U1/Z -to U2/A
```

The following example annotates a rise net delay of 12.3 for minimum delay analysis between output pin U1/Z and input pin U2/A. In this case, the net delay value does include load delay.

```
prompt> set_annotated_delay -net -rise -min -load_delay net 12.3 \
      -from U1/Z -to U2/A
```

SEE ALSO

- current_design(2)
- remove_annotated_delay(2)
- report_timing(2)
- report_annotated_delay(2)
- reset_design(2)
- set_input_delay(2)
- set_dont_touch(2)
- report_dont_touch(2)
- target_library(3)

set_annotated_transition

Sets the transition time at a given pin.

SYNTAX

```
status set_annotated_transition
[-rise | -fall]
[-min]
[-max]
transition
port_pin_list
```

Data Types

```
transition    float
port_pin_list list
```

ARGUMENTS

-rise | -fall

Specifies whether the transition time is for a data rise or data fall transition. If you do not specify **-rise** or **-fall**, both values are set.

-min

Specifies that the transition is to be used for minimum delay analysis. By default, the transition value is used for maximum and minimum delay analysis. If a transition value is annotated for only minimum or maximum delay analysis, that value is used for both maximum and minimum delays. It is possible to annotate different transition values for minimum and maximum analysis, but it is not possible to annotate only for minimum or annotate only for maximum.

-max

Specifies that the transition is to be used for maximum delay analysis. By default, the transition value is used for maximum and minimum delay analysis. If a transition value is annotated for only minimum or maximum delay analysis, that value is used for both maximum and minimum delays. It is possible to annotate different transition values for minimum and maximum analysis, but it is not possible to annotate only for minimum or annotate only for maximum.

transition

Specifies the transition value at the pins supplied with the *port_pin_list* argument. The transition value must be expressed in units consistent with the technology library used during optimization. For example, if the technology library specifies transition values in nanoseconds, the transition values must be expressed in nanoseconds.

port_pin_list

Specifies a list of leaf-cell pins or top-level ports that are the endpoints of the timing arcs for which delays are to be annotated.

DESCRIPTION

This command sets the transition time at a given pin.

To list annotated transition values, use the **report_annotated_transition** command. To remove the annotated transition values from a design, use the **remove_annotated_transition** or **reset_design** command.

Multicorner-Multimode Support

This command applies to current scenario only.

EXAMPLES

The following example annotates a transition time of 20 units at input pin *A* of cell instance U1/U2/U3:

```
prompt> set_annotated_transition 20 U1/U2/U3/A
```

The following example annotates a rise transition of 1.4 units at the input pin U5/A:

```
prompt> set_annotated_transition -rise 1.4 U5/A
```

The following example annotates a rise transition of 12.3 units for minimum delay analysis at U5/A:

```
prompt> set_annotated_transition -rise -min 12.3 U5/A
```

SEE ALSO

- current_design(2)
- link(2)
- remove_annotated_transition(2)
- report_annotated_transition(2)
- report_timing(2)
- reset_design(2)
- set_input_transition(2)
- target_library(3)

set_app_var

Sets the value of an application variable.

SYNTAX

```
string set_app_var  
-default  
  var  
  value
```

Data Types

<i>var</i>	string
<i>value</i>	string

ARGUMENTS

-default

Resets the variable to its default value.

var

Specifies the application variable to set.

value

Specifies the value to which the variable is to be set.

DESCRIPTION

The **set_app_var** command sets the specified application variable. This command sets the variable to its default value or to a new value you specify.

This command returns the new value of the variable if setting the variable was successful. If the application variable could not be set, then an error is returned indicating the reason for the failure.

Reasons for failure include:

- The specified variable name is not an application variable, unless the application variable **sh_allow_tcl_with_set_app_var** is set to true See the **sh_allow_tcl_with_set_app_var** man page for details.

- The specified application variable is read only.
- The value specified is not a legal value for this application variable

EXAMPLES

The following example attempts to set a read-only application variable:

```
prompt> set_app_var synopsis_root /tmp  
Error: can't set "synopsys_root": variable is read-only  
Use error_info for more info. (CMD-013)
```

In this example, the application variable name is entered incorrectly, which generates an error message:

```
prompt> set_app_var sh_enabel_page_mode 1  
Error: "sh_enabel_page_mode" is not an application variable  
Use error_info for more info. (CMD-013)
```

This example shows the variable name entered correctly:

```
prompt> set_app_var sh_enable_page_mode 1  
1
```

This example resets the variable to its default value:

```
prompt> set_app_var sh_enable_page_mode -default  
0
```

SEE ALSO

get_app_var(2)
report_app_var(2)
write_app_var(2)

set_aspect_ratio

Specifies the placement aspect ratio for the core area. This command is supported only in topographical mode.

SYNTAX

```
status set_aspect_ratio  
      y_x_ratio
```

Data Types

```
y_x_ratio    float
```

ARGUMENTS

y_x_ratio

Specifies the aspect ratio constraint for the core area; it is defined as y:x ratio. For example, specifying 1.2 as the aspect ratio results in a core area whose y dimension is 1.2 times the x dimension. The default value for the aspect ratio is 1.0.

DESCRIPTION

The **set_aspect_ratio** command specifies the design-level physical constraint for the aspect ratio, which is used to generate the coarse floorplan during synthesis. Use this command before running synthesis commands, and after loading the physical library and the design.

The other commands that set constraints on the core area, are **set_placement_area**, **set_rectilinear_outline**, and **set_aspect_ratio**. Among the four commands, **set_placement_area** has the highest priority, while **set_rectilinear_outline** has higher priority than **set_utilization** and **set_aspect_ratio**. This means if the placement area constraint is set by **set_placement_area** or if the rectilinear outline constraint is set by **set_rectilinear_outline** on the current design, then the aspect ratio constraint will not be used during synthesis. Furthermore, in this case, the aspect ratio will neither be reported by the command **report_physical_constraints**, nor written out by the **write_physical_constraints** command.

EXAMPLES

The following example shows how to set the aspect ratio to 2.0:

```
prompt> set_aspect_ratio 2.0
```

SEE ALSO

report_physical_constraints(2)
set_placement_area(2)
write_physical_constraints(2)

set_attribute

Sets an attribute to a specified value on the specified list of objects.

SYNTAX

```
collection set_attribute  
  objects  
  attribute_name  
  attribute_value  
  [-type boolean | integer | float | string]  
  [-bus]  
  [-quiet]
```

Data Types

<i>objects</i>	collection
<i>attribute_name</i>	string
<i>attribute_value</i>	string

ARGUMENTS

objects

Specifies the objects on which the attribute is to be set.

attribute_name

Specifies the name of the attribute to be set.

attribute_value

Specifies the value of the attribute. The data type must be the same as that of the attribute.

-type boolean | integer | float | string

Specifies the data type of the *attribute_value* argument. This argument is required when creating new attributes; otherwise, it is optional. If the attribute data type is not specified, the tool uses either the specified *attribute_value* argument or the data type of the existing attribute.

-bus

Sets attributes on the bus instead of bus members.

-quiet

Turns off the warning message that would otherwise be issued if the attribute or objects are not found.

DESCRIPTION

This command sets the value of an attribute on an object. For a complete list of attributes, see the **attributes** man page.

This command returns a collection of objects that have the specified attribute value set. If the attribute is not set on any objects, the command returns an empty string.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information if there is no scenario dependency for the attributes. However, for attributes such as **max_leakage_power**, which are scenario dependent, this command uses information from the current scenario only.

EXAMPLES

The following example defines an integer cell attribute named X, and then sets this attribute to 30 on all cells in this level of the hierarchy:

```
prompt> define_user_attribute -type int -classes cell X
cell
prompt> set_attribute [get_cells *] X 30
{U1}
```

SEE ALSO

collections(2)
define_user_attribute(2)
get_attribute(2)
list_attributes(2)
remove_attribute(2)
attributes(3)

set_auto_disable_drc_nets

Sets the **auto_disable_drc_net** attribute on the current design, causing the specified nets to have DRC disabled. This command was previously called by the **set_auto_ideal_nets** command.

SYNTAX

```
status set_auto_disable_drc_nets
  [-default]
  [-none]
  [-all]
  [-clock true | false]
  [-constant true | false]
  [-scan true | false]
  [-on_clock_network true | false]
```

ARGUMENTS

-default

Disables design rule checking (DRC) on nets connected to clocks and constant nets in the current design. This option is equivalent to **-clock true** and **-constant true**. This is also the default behavior if no option is selected or if the command is not executed. You cannot use **-default** with any other option.

-none

Enables DRC for all nets in the current design, including nets connected to clocks. You cannot use **-none** with any other option.

-all

Disables DRC on all applicable nets in the current design. It is equivalent to setting **-clock true**, **-constant true**, and **-scan true**.

-clock true | false

Disables or enables DRC on nets that are connected to clock pins of sequential cells for ideal clocks with a user-specified clock transition time. When set to **false**, DRC is enabled on these nets. This option is supported for backward compatibility. However, it is recommended that you use the **-on_clock_network** option instead to enable or disable DRC for the clock network.

-constant true | false

Disables or enables DRC on constant nets in the current design. When set to **true** (the default), it automatically disables DRC on constant nets. When set to **false**, it enables DRC on constant nets.

-scan true | false

Disables or enables DRC on scan clock and scan enable nets in the current design. When set to **true**, it automatically disables DRC on scan clock and scan enable nets. When **false** (the default), it enables DRC on scan clock and scan enable nets. This

option does not prevent the tool from automatically disabling DRC on nets connected to clocks and constant nets, unless **-clock false** and **-constant false** is also used. Scan data nets (those specified in the SCANDEF) are not affected by this option.

-on_clock_network true | false

Disables or enables DRC on clock networks in the current design. When set to **true**, DRC is disabled on clock networks. The default is **false**. It is recommended that you use this option instead of the **-clock** option.

DESCRIPTION

This command disables design rule checking (DRC) for specified nets in the current design. If the command is executed without options, or if the command is not executed at all, by default, the tool disables DRC for all nets connected to clocks and constant nets. The command options are additive each time a new **set_auto_disable_drc_nets** command is executed.

To disable DRC for clock networks, use the **-on_clock_network true** option. The **-clock true** option is kept only for backward compatibility.

To prevent constant nets from having DRC disabled, use the **-constant false** or **-none** options.

DRC disabled nets are those nets that are free from the max_capacitance, max_fanout, and max_transition design rule constraints. They are useful for reducing DRC violations caused by clock trees, because these nets usually have high max_capacitance and max_fanout violations.

You can add the **dont_touch** attribute to the DRC disabled nets by using the **set_dont_touch_network** or the **set_dont_touch** command.

Note that nets with the **auto_disable_drc_nets** attribute are different from nets with the **ideal_nets** and **ideal_networks** attributes because the nets with the **ideal_nets** and **ideal_networks** attributes use ideal timing and the **dont_touch** attribute.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example disables DRC on nets connected to clocks and constant nets in the current design:

```
prompt> set_auto_disable_drc_nets -clock true -constant true
```

The following example disables DRC on only the constant nets in the current design:

```
prompt> set_auto_disable_drc_nets -constant true
```

The following example re-enables DRC for all relevant nets, including those on clock nets, in the current design:

```
prompt> set_auto_disable_drc_nets -none
```

SEE ALSO

report_attribute(2)
report_compile_options(2)
report_design(2)
reset_design(2)
set_dont_touch(2)
set_dont_touch_network(2)
set_ideal_net(2)
set_ideal_network(2)
set_clock_transition(2)

set_auto_ideal_nets

This command is obsolete. It has been replaced by the **set_auto_disable_drc_nets** command, which works in the same way. The change was made to correctly distinguish between DRC disabled nets and ideal nets, which additionally have ideal timing and dont_touch.

SEE ALSO

set_auto_disable_drc_nets(2)
set_ideal_net(2)
set_ideal_network(2)

set_autofix_configuration

Configures automatic fixing of violations (AutoFix), performed by DFT insertion, globally for the current design.

SYNTAX

```
status set_autofix_configuration
  [-type fix_type]
  [-control_signal control_name]
  [-test_data test_data_name]
  [-method method]
  [-fix_data enable | disable]
  [-fix_latch enable | disable]
  [-include_elements list_of_design_objects]
  [-exclude_elements list_of_design_objects]
```

Data Types

```
fix_type           string
control_name       string
test_data_name     string
method             string
list_of_design_objects list
```

ARGUMENTS

-type *fix_type*

Specifies the type of fixing to configure. For multiple fixing types, specify a configuration command for each type. The valid *fix_type* values are:

```
clock
set
reset
xpropagation
internal_bus
external_bus
bidirectional
```

-control_signal *control_name*

Specifies the name of the control signal that activates the fixing logic. The specified signal must be predefined as either a **TestMode** or **ScanEnable** signal. The **ScanEnable** signal type is valid only for the **set** and **reset** fixing types when using the **gate\gP** fixing method. **If no control signal is specified, the AutoFix test points are controlled by any signal with the TestMode signal type. If no TestMode signal is found, Autofix creates a new test mode port.**

-test_data *test_data_name*

Specifies the name of the test data signal used for fixing violations. This signal is used as a controllable source signal by the AutoFix test points. The signal must be predefined using the **set_dft_signal** command as a **TestData** signal. In addition, the signal must also be predefined as a **ScanClock** signal type for the **clock** fixing type, or as a **Reset** signal type for the **set** and **reset** fixing types. If no test data signal is specified, Autofix creates a new **ScanClock** or **Reset** port depending on the fixing type.

-method *method*

Specifies the fixing method to use for the fixing type specified with the **-type** option. Valid methods depend on the fixing type, as follows:

clock - mux
set - mux | gate (default is mux)
reset - mux | gate (default is mux)
xpropagation - mux
internal_bus - no_disabling | enable_one | disable_all (default is enable_one)
external_bus - no_disabling | enable_one | disable_all (default is disable_all)
bidirectional - input | output | no_disabling (default is input)

-fix_data enable | disable

Enables or disables fixing of clock-used-as-data and reset-as-data violations. This option is only valid for the **clock**, **set**, and **reset** fixing types. The default is **disable**.

-fix_latch enable | disable

Enables or disables fixing of uncontrollable set and reset pins on latches. The default is **disable**. This option is only valid for the **set** and **reset** fixing types. This option does not affect the enable signal of the latch.

-include_elements *list_of_design_objects*

Specifies the list of design objects to which the global fixing configuration applies. Wildcards and collections are supported.

If this option is used, only the specified design objects are fixed. By default, all violated design objects are AutoFixed.

Valid object types depend on the fixing type, as follows:

clock - cell (hierarchical and leaf)
set - cell (hierarchical and leaf)
reset - cell (hierarchical and leaf)
xpropagation - cell (hierarchical and leaf)
internal_bus - net
external_bus - net
bidirectional - port

-exclude_elements *list_of_design_objects*

Specifies the list of design objects to exclude from global fixing. Wildcards and collections are supported. By default, the list of excluded design objects is empty.

Valid object types are the same as for the **-include_elements** option.

DESCRIPTION

The **set_autofix_configuration** command allows you to specify details about how different DFT violation types should be AutoFixed. This command specifies a global fixing configuration for the current design. To specify different fixing configurations for different

parts of the design, use the **set_autofix_element** command instead.

Note that this command does not enable or disable AutoFix fixing types. Before configuring fixing with this command, you must enable AutoFix by using the **-fix_clock**, **-fix_set**, **-fix_reset**, **-fix_xpropagation**, **-fix_bus**, or **-fix_bidirectional** options of the **set_dft_configuration** command. See that man page for more details.

A separate configuration is maintained for each fixing type. If multiple **set_autofix_configuration** commands are issued for the same fixing type, the most recent fixing specification for that type replaces previous specifications.

The AutoFix configuration is applied to the current design. If you apply an Autofix configuration and then change the current design, your settings are no longer visible.

Use the **report_autofix_configuration** command to display the current Autofix configuration for the design. Use the **reset_autofix_configuration** command to reset the Autofix configuration for the design.

EXAMPLES

In the following example, AutoFix is configured to fix uncontrollable clock violations using the TESTCLK signal as the new Autofix clock signal, and the TM_FIX signal as the AutoFix fixing logic enable pin. Note that the test mode and controllable clock pin signals are defined with the **set_dft_signal** command.

```
prompt> set_dft_signal -view existing_dft -type ScanClock \  
-timing {45 55} -port TESTCLK  
  
prompt> set_dft_signal -view spec -type TestMode -port TM_FIX  
  
prompt> set_dft_configuration -fix_clock enable  
  
prompt> set_dft_signal -view spec -type TestData -port TESTCLK  
  
prompt> set_autofix_configuration -type clock \  
-include_elements [get_object_name [get_cells -hierarchical *]] \  
-control_signal TM_FIX \  
-test_data TESTCLK
```

SEE ALSO

set_dft_configuration(2)
report_autofix_configuration(2)
reset_autofix_configuration(2)
preview_dft(2)
insert_dft(2)
set_dft_signal(2)

set_autofix_element

Configures automatic fixing of violations (AutoFix), performed by DFT insertion, locally for a particular part of the design.

SYNTAX

```
status set_autofix_element
    list_of_design_objects
    [-type fix_type]
    [-control_signal control_name]
    [-test_data test_data_name]
    [-method method]
    [-fix_data enable | disable]
    [-fix_latch enable | disable]
```

Data Types

```
list_of_design_objects  list
fix_type               string
control_name           string
test_data_name        string
method                 string
```

ARGUMENTS

list_of_design_objects

Specifies the list of design objects to which the local fixing configuration applies. Wildcards and collections are not supported.

Valid object types depend on the fixing type, as follows:

```
clock - cell (hierarchical and leaf)
set - cell (hierarchical and leaf)
reset - cell (hierarchical and leaf)
xpropagation - cell (hierarchical and leaf)
internal_bus - net
external_bus - net
bidirectional - port
```

-type *fix_type*

Specifies the type of fixing to configure. For multiple fixing types, specify a configuration command for each type. The valid *fix_type* values are:

```
clock
set
```

reset
xpropagation
internal_bus
external_bus
bidirectional

-control_signal *control_name*

Specifies the name of the control signal that activates the fixing logic. The specified signal must be predefined as either a **TestMode** or **ScanEnable** signal. The **ScanEnable** signal type is valid only for the **set** and **reset** fixing types. If no control signal is specified, the AutoFix test points are controlled by any signal with the **TestMode** signal type. If no **TestMode** signal is found, Autofix creates a new test mode port.

-test_data *test_data_name*

Specifies the name of the test data signal used for fixing violations. This signal is used as a controllable source signal by the AutoFix test points. The signal must be predefined using the **set_dft_signal** command as a **TestData** signal. In addition, the signal must also be predefined as a **ScanClock** signal type for the **clock** fixing type, or as a **Reset** signal type for the **set** and **reset** fixing types. If no test data signal is specified, Autofix creates a new **ScanClock** or **Reset** port depending on the fixing type.

-method *method*

Specifies the fixing method to use for the fixing type specified with the **-type** option. Valid methods depend on the fixing type, as follows:

clock - **mux**
set - **mux** | **gate** (default is **mux**)
reset - **mux** | **gate** (default is **mux**)
xpropagation - **mux**
internal_bus - **no_disabling** | **enable_one** | **disable_all** (default is **enable_one**)
external_bus - **no_disabling** | **enable_one** | **disable_all** (default is **disable_all**)
bidirectional - **input** | **output** | **no_disabling** (default is **input**)

-fix_data **enable** | **disable**

Enables or disables fixing of clock-used-as-data and reset-as-data violations. This option is only valid for the **clock**, **set**, and **reset** fixing types. The default is **disable**.

-fix_latch **enable** | **disable**

Enables or disables fixing of uncontrollable set and reset pins on latches. The default is **disable**. This option is only valid for the **set** and **reset** fixing types. This option does not affect the enable signal of the latch.

DESCRIPTION

The **set_autofix_element** command allows you to specify details about how different DFT violation types should be AutoFixed in a particular part of the design. This command specifies a local fixing configuration that overrides the global fixing configuration specified by the **set_autofix_configuration** command.

Note that this command does not enable or disable AutoFix fixing types. Before configuring fixing with this command, you must enable AutoFix by using the **-fix_clock**, **-fix_set**, **-fix_reset**, **-fix_xpropagation**, **-fix_bus**, or **-fix_bidirectional** options of the **set_dft_configuration** command. See that man page for more details.

Separate configurations are maintained for each fixing type. If multiple **set_autofix_element** commands are issued for the same fixing type on the same design object, the most recent fixing specification for that type replaces previous specifications.

The AutoFix configuration is applied to the current design. If you apply an Autofix configuration and then change the current design, your settings are no longer visible.

Use the **report_autofix_element** command to display the current Autofix configuration for the design. Use the **reset_autofix_element** command to reset the Autofix configuration for the design.

EXAMPLES

The following example configures clock-fixing globally for the design using CLK, then configures clock-fixing locally in hierarchical cells U_RX_BLK and U_TX_BLK:

```
prompt> set_autofix_configuration -type clock -test_data CLK
```

```
prompt> set_autofix_element -type clock -test_data RXCLK {U_RX_BLK}
```

```
prompt> set_autofix_element -type clock -test_data TXCLK {U_TX_BLK}
```

SEE ALSO

- current_design(2)
- insert_dft(2)
- preview_dft(2)
- report_autofix_configuration(2)
- report_autofix_element(2)
- set_autofix_configuration(2)

set_balance_registers

Sets the **balance_registers** attribute on the specified designs or on the current design, so that the design is retimed during compile.

SYNTAX

```
status set_balance_registers  
[-design]  
[true | false]  
[design_list]
```

Data Types

design_list list

ARGUMENTS

-design

Specifies a design on which to set the **balance_registers** attribute.

true | false

Specifies the value with which to set the **balance_registers** attribute. The default value is **true**, which allows the **compile** command to automatically invoke the **balance_registers** command. When the value is set to **false**, **balance_registers** is not invoked.

design_list

Specifies a list of designs to retime. The default is the current design.

DESCRIPTION

This command sets the **balance_registers** attribute on the specified designs or on the current design, so that the design is retimed during compile. If the **balance_registers** attribute is set to **true** (the default) on a design, the **compile** command automatically invokes the **balance_registers** command, which moves registers to minimize the maximum register-to-register delay. Subdesigns in the hierarchy are ungrouped into the design, unless the **dont_touch** attribute is set.

It is an error to invoke **balance_registers** on a design that contains generic logic. If the **balance_registers** attribute is set, **compile** attempts to optimize the design by invoking **balance_registers**. Ensure that your design contains no generic logic at the instant **balance_registers** is called during **compile**.

For more details on retiming during optimization, see the Design Compiler documentation.

To remove **balance_registers**, use the **remove_attribute** or **reset_design** command. You can achieve the same effect by setting the **balance_registers** attribute to **false** by running **set_balance_registers false**.

Licensing during compile when using the **balance_registers** attribute can be controlled by using the **compile_retime_license_behavior** shell variable.

EXAMPLES

In the following example, **balance_registers** is enabled on the design named TEST:

```
prompt> set_balance_registers -design TEST
```

In the following example, **balance_registers** is enabled on the current design and **balance_registers** is disabled on the design named OLD:

```
prompt> read TEST
```

```
prompt> set_balance_registers
```

```
prompt> set_balance_registers false -design OLD
```

SEE ALSO

- balance_registers(2)
- compile(2)
- current_design(2)
- remove_attribute(2)
- reset_design(2)
- attributes(3)

set_boundary_cell

Sets the boundary-cell configuration for the specified ports and core cells.

SYNTAX

```
status set_boundary_cell
  -class core_wrapper | shadow_wrapper | bsd
  [-function function]
  [-ports port_list]
  [-core_cells core_list]
  [-type type]
  [-design design_name]
  [-register_io_implementation swap | in_place]
  [-use_dedicated_wrapper_clock true | false]
  [-safe_state 0 | 1 | none]
  [-share true | false]
  [-name bcell_name]
  [-shift_clk bcell_shift_clk]
  [-instance hier_cell_list]
  [-include cell_list]
  [-exclude cell_list]
  [-reuse_threshold threshold_value]
  [-hookup_pin hookup_pin_name]
  [-pins list_of_pins]
```

Data Types

<i>function</i>	string
<i>port_list</i>	list
<i>core_list</i>	list
<i>type</i>	string
<i>design_name</i>	string
<i>bcell_name</i>	string
<i>bcell_shift_clk</i>	string
<i>hier_cell_list</i>	list
<i>cell_list</i>	list
<i>threshold_value</i>	integer
<i>hookup_pin_name</i>	string

ARGUMENTS

-class *core_wrapper* | *shadow_wrapper* | *bsd*

Specifies the name of the class for which the configuration applies. Valid values are **core_wrapper**, **shadow_wrapper**, and **bsd**.

This argument is required.

-function *function*

Specifies the function of the specified ports to which the local configuration applies. There is no default value for this option. The following values are valid for *function*:

input
output
control
bidir
observe
input_inverted
output_inverted
bidir_inverted
receiver_p
receiver_n
ac_select

This option is only supported when "-class bsd" is specified.

-ports *port_list*

Specifies the list of ports for which the configuration applies. There is no default value for this option. You must specify either the **-ports** option or the **-core** option.

-core_cells *core_list*

Specifies a list of core cells to which the boundary configuration applies.

-type *type*

Specifies the cell type to be used for the boundary cell in DFT insertion. There is no default value for this option. If this option is not specified, the tool uses the **set_wrapper_configuration** command to get the type of DFT design to use for the specified ports. The following values are valid for *type*:

WC_D1
WC_D1_S
WC_S1
WC_S1_S
BC_1
BC_2
BC_4
BC_7
BC_8
BC_9
AC_1
AC_2
AC_7
AC_SELU
AC_SELX
none

-design *design_name*

Specifies the name of the user-defined DFT design as specified by the **define_dft_design** command to be used for the boundary cell.

There is no default value for this option. By default, the tool uses a DesignWare cell for DFT insertion.

This option is mutually exclusive with the **-type** option. You must specify either the **-type** option or the **-design** option.

-register_io_implementation swap | in_place

Specifies the implementation to use for synthesizing shared wrapper cells in core wrapping the current design.

When the option is set to **swap**, registers connected to the specified ports of the design are replaced with equivalent shared wrapper cells. When the option is set to **in_place**, additional glue logic is added to registers attached to the specified ports of the design to reuse them in core wrapper chains.

There is no default value for this option. If the option is not specified, the tool uses the **set_wrapper_configuration** command to get the implementation style to use for the specified ports.

-use_dedicated_wrapper_clock true | false

Specifies whether the dedicated wrapper clock is used for shared wrapper. When the option is set to **true**, core wrapper application uses the dedicated wrapper clock for all shared wrapper cells of the specified ports of the design. When the option is set to **false**, functional clocks are used for shared wrapper cells connected to the specified ports of the design.

-safe_state 0 | 1 | none

Specifies the safe state to be used for wrapper cells added by core wrapping. Valid values for this option are **0**, **1**, and **none**.

There is no default value for this option. If the option is not specified, the tool uses the **set_wrapper_configuration** command to get the safe state to use for the specified ports.

-share true | false

Specifies whether the boundary cells should be shared. Sharing boundary cells is allowed only for control boundary cells. If the value of the option is set to **true**, the boundary cells for the specified ports are shared. If the value of the option is set to **false**, the boundary cells for the specified ports are not shared.

The default value of this option is **true** for function value control and **false** for all other types of boundary cells.

-name bcell_name

Specifies the name of the boundary cell. The name can be subsequently referenced in a **set_scan_path** specification to control the boundary cell ordering.

This option is mandatory if the **-share** option is set to **true** or if the **-function** option is set to **control** or **ac_select**.

-shift_clk bcell_shift_clk

Specifies the shift clock to be used for the boundary cell.

This option is applicable only with the **-class core_wrapper** and either of the **-type WC_D1** or **-type WC_D1_S** options.

The value of this option must be a port or an internal pin. The specified port or pin is connected to the shift clock pin of the dedicated boundary cell.

The timing information of the specified clock is obtained from DRC.

There is no default value for this option.

-instance hier_cell_list

This option is applicable only when the **-class** option is set to **core_wrapper** and the **-maximize_reuse** option of the **set_wrapper_configuration** command is enabled.

This option specifies a list of hierarchical cells. If a port is connected to a register inside one of these hierarchical cells, the tool adds the specified type of boundary cell to the port.

The valid types of boundary cells supported with this option are **none** or **WC_D1**.

If the type of the boundary cell is chosen as **none**, no wrapper cell is added to the port.

If the type of the boundary cell is chosen as **WC_D1**, a dedicated wrapper cell is added to the port.

This option is mutually exclusive with the **-port** option.

There is no default value for this option.

-include *cell_list*

Applies only when the **-class** option is set to **core_wrapper** and the **-maximize_reuse** option of the **set_wrapper_configuration** command is enabled.

This option specifies the list of I/O register cells to be included for the specified port.

There is no default value for this option.

-exclude *cell_list*

Applies only when the class option is set to **core_wrapper** and the **-maximize_reuse** option of the **set_wrapper_configuration** command is enabled.

This option specifies the list of I/O register cells to be excluded for the specified port.

To specify a list of cells to be excluded from all the ports, do not specify the port names with the command.

There is no default value for this option.

-reuse_threshold *threshold_value*

Specifies the maximum number of I/O registers that can be reused as shared wrapper cells for the specified ports.

This option is applicable only when the **-class** option is set to **core_wrapper**.

This option is applicable only when the **-maximize_reuse** option of the **set_wrapper_configuration** command is enabled.

When the number of I/O registers detected for the specified ports exceeds the specified threshold value, a dedicated wrapper cell is added to the ports.

The default value of this option is -1; that is, the reuse threshold value specified with wrapper configuration is applicable to the specified ports.

If the value of the option is set to 0, all I/O registers associated with the specified ports are reused as shared wrapper cells.

-hookup_pin *hookup_pin_name*

Specifies the name of the hookup pin to be used to stitch custom BSR cells. When the **-hookup_pin** option is not specified, the BSR cells are hooked up at pad pins.

This option is applicable only when the **-class** option is set to **bsd** and the **-function** option is set to **input**, **output**, or **control**.

DESCRIPTION

The **set_boundary_cell** command overrides the default configuration for the specified ports and core cells of the current design. The ports and core cells must exist in the current design.

To set the default core wrapper configuration, use the **set_wrapper_configuration -class core_wrapper** command.

To set the default shadow wrapper configuration, use the **set_wrapper_configuration -class shadow_wrapper** command.

To set the default BSD configuration, use the **set_bsd_configuration** command.

EXAMPLES

The following example specifies that the DesignWare WC_D1 cell is to be used with safe value of 1 on *port1* and *port2* in core wrapping:

```
prompt> set_boundary_cell -class core_wrapper -type WC_D1 \
    -safe_state 1 -port_list {port1 port2}
```

The following example specifies that the DesignWare BC_1 cell is to be used for ports *port1* and *port2* in BSD insertion:

```
prompt> set_boundary_cell -class bsd -type BC_1 \
    -function output -port_list {port1 port2}
```

The following example specifies that the DesignWare BC_2 cell is to be used as a control BSR cell for ports *port1* and *port2* in BSD insertion: The control BSR cells for these ports are not shared.

```
prompt> set_boundary_cell -class bsd -type BC_2 \
    -function control -port_list {port1 port2} \
    -name CTRL1 -share false
```

The following example specifies that the DesignWare BC_2 cell is to be used as a control BSR cell for ports *port1* and *port2* in BSD insertion. The control BSR cell is shared for these ports.

```
prompt> set_boundary_cell -class bsd -type BC_2 \
    -function control -port_list {port1 port2} \
    -name CTRL1 -share true
```

The following example specifies that the DesignWare AC_SELU cell is to be used as the AC selection BSR cell for ports *ac_port1* and *ac_port2* in BSD insertion. The AC selection BSR cell is shared for these ports.

```
prompt> set_boundary_cell -class bsd -type AC_SELU \
    -function ac_select -port_list {ac_port1, ac_port2} \
    -name SELECT1
```

The following example specifies a user defined implementation of BC_1 BSR cell to be used for ports *port1* and *port2* in BSD insertion:

```
prompt> define_dft_design -type BC_1 -design my_bc_1 \
    -interface { capture_clk cclk h \
    update_clk uclk h shift_dr se h mode tm h si si h \
    data_in di h data_out do h so so h }

    set_boundary_cell -class bsd -design my_bc_1 -port_list {port1 port2}
```

The following example specifies that no boundary cell to be added to the ports connected to hierarchical cell *core1_inst/core_inst*:

```
prompt> set_boundary_cell -class core_wrapper -type none \
    -instance core1_inst/core2_inst
```

The following example specifies that a dedicated boundary cell to be added to the ports connected to hierarchical cell *core1_inst/core_inst*.

```
prompt> set_boundary_cell -class core_wrapper -type WC_D1 \
```

-instance core1_inst/core2_inst

The following example specifies that l2_reg, l2_1_reg to be included as boundary cells for the port l2:

```
prompt> set_boundary_cell -class core_wrapper -port { l2 } \
      -include { l2_reg l2_1_reg }
```

The following example specifies that l2_reg, l2_1_reg to be excluded as boundary cells for the port l2:

```
prompt> set_boundary_cell -class core_wrapper -port { l2 } \
      -exclude { l2_reg l2_1_reg }
```

The following example specifies that the custom BSR minimp_o_bc1, minimp_bc4 and minimp_e_bc1 should be stitched to the specified hookup pins.

```
prompt> set_boundary_cell -class bsd -design minimp_o_bc1 \
      -function output -port DDR_UDQS -hookup_pin i_phy_64b/bs_dqs[1]
prompt> set_boundary_cell -class bsd -design minimp_bc4 \
      -function input -port DDR_UDQS -hookup_pin i_phy_64b/bs_dqs_in_observe[1]
prompt> set_boundary_cell -class bsd -design minimp_e_bc1 \
      -function control -port DDR_UDQS -name bc_e_DDR_UDQS \
      -hookup_pin i_phy_64b/bs_dqsoe[1]
```

The following example specifies that l2_reg, l2_1_reg to be excluded as boundary cells for the design:

```
prompt> set_boundary_cell -class core_wrapper \
      -include { l2_reg l2_1_reg }
```

SEE ALSO

```
insert_dft(2)
preview_dft(2)
remove_boundary_cell(2)
report_boundary_cell(2)
set_bsd_configuration(2)
set_wrapper_configuration(2)
```

set_boundary_cell_io

Specifies the primary inputs (PIs) and primary outputs (POs) of a boundary cell in the current design.

SYNTAX

```
status set_boundary_cell_io  
-access {pin_type pin_name}  
-type boundary | wrapper  
-cell shift_flop_name
```

Data Types

<i>pin_type</i>	string
<i>pin_name</i>	string
<i>shift_flop_name</i>	string

ARGUMENTS

-access {*pin_type pin_name*}

Specifies a list of pairs, with each pair consisting of the pin type and the hierarchical pin name. Valid values for *pin_type* are **in_pi**, **in_po**, **out_pi**, **out_po**.

-type boundary | wrapper

Specifies the type of cell as either **boundary** or **wrapper**.

-cell *shift_flop_name*

Specifies the names of the shift flop of the boundary cell.

DESCRIPTION

The **set_boundary_cell_io** command specifies the primary inputs (PIs) and primary outputs (POs) of boundary cells. For input cells, specify only **in_pi** and **in_po**. For output and control cells, specify **out_pi** and **out_po**. For merged cells, specify all four values (**in_pi**, **in_po**, **out_pi**, **out_po**). The cell name indicates the name of the shift flop of the cell.

EXAMPLES

set_boundary_cell_io

2250

The following example instructs **check_bsd** to use the specified primary inputs and primary outputs for the BSR cell *bs_d_oe2/q2bscan/inst1/inst4*:

```
prompt> set_boundary_cell_io -type boundary \  
-access {out_pi bs_d_oe2/U1/**logic_0** out_po U8/U1/A} \  
-cell bs_d_oe2/q2bscan/inst1/inst4/q_reg \  

```

SEE ALSO

check_bsd(2)
write_bsd(2)

set_boundary_optimization

Sets the **boundary_optimization** attribute on specified cells, references, or designs, thus allowing for optimization across hierarchical boundaries.

SYNTAX

```
status set_boundary_optimization  
      object_list  
      [true | false]
```

Data Types

object_list list

ARGUMENTS

object_list

Specifies a list of cell, reference, or design names for which to enable boundary optimization. Cell or reference names in *object_list* must be from the current design. If more than one object is specified, they must be enclosed in quotation marks (") or braces ({}).

true | false

Specifies the value with which to set the **boundary_optimization** attribute. When this option is set to **true** (the default), boundary optimization is enabled.

DESCRIPTION

This command sets the **boundary_optimization** attribute on *object_list*. This attribute is used to specify the cells, references, or designs to be optimized across hierarchical boundaries. The compile commands use this information to create smaller designs. This might change the function of the object, so the object must not be used in any other context.

If a cell with a specified name is found in the current design, the **boundary_optimization** attribute is set to the specified value in the cell. If no cell with a specified name is found, the tool searches for a reference. If a reference is found, the attribute is set for the reference. Otherwise, the tool searches for a design and the attribute is set for the design.

By default, setting the **boundary_optimization** attribute to **false** does not disable constant propagation across hierarchical boundaries of the specified object. To disable constant propagation with the **boundary_optimization false** attribute, set the **compile_enable_constant_propagation_with_no_boundary_opt** variable to **false**.

Occasionally, cells use an input signal only in its complemented form. In this case, it is best to invert the signal and its ports. The

set_boundary_optimization command considers these optimizations. When this occurs, port names are changed according to the **port_complement_naming_style** variable.

To remove this attribute, use the **remove_attribute** command.

EXAMPLES

The following example shows how to ignore hierarchical boundaries during optimization for cells named U0 and U1 and how to preserve them for a cell named U2:

```
prompt> set_boundary_optimization { U0 U1 }
```

```
prompt> set_boundary_optimization U2 false
```

SEE ALSO

- compile(2)
- find(2)
- get_attribute(2)
- remove_attribute(2)
- reset_design(2)
- uniquify(2)
- port_complement_naming_style(3)

set_bsd_ac_port

Identifies the AC ports in your design.

SYNTAX

```
status set_bsd_ac_port  
-port_list list_of_ports
```

Data Types

list_of_ports list

ARGUMENTS

-port_list *list_of_ports*

Specifies the ports of the current design to be considered as AC ports.

DESCRIPTION

The **set_bsd_ac_port** command identifies the ports of the current design to be considered as AC ports. Boundary Scan Insertion adds AC BSR cells for such ports if the IEEE1149.6_2003 standard mode is enabled.

This specification is honored only by Boundary Scan Synthesis.

To report the defined AC ports, use the **report_bsd_ac_port** command. To delete AC port specifications, use the **remove_bsd_ac_port** command.

EXAMPLES

The following example shows how the ports A_1, A_2, and A_3 are identified as AC ports:

```
prompt> set_bsd_ac_port -port_list {A_1 A_2 A_3}
```

SEE ALSO

insert_dft(2)
preview_dft(2)
remove_bsd_ac_port(2)
report_bsd_ac_port(2)

set_bsd_compliance

Specifies an IEEE 1149.1 compliant pattern for a boundary-scan design.

SYNTAX

```
status set_bsd_compliance  
-name pattern_name  
-pattern signal_port_bit_value_pairs
```

Data Types

```
pattern_name          string  
signal_port_bit_value_pairs list
```

ARGUMENTS

-name *pattern_name*

Specifies the name of the pattern. Patterns having the same name merge and overwrite earlier versions.

-pattern *signal_port_bit_value_pairs*

Lists signal port and binary value pairs that specify the compliance-enable pattern. The specified signal ports must be input ports and cannot include IEEE 1149.1 test access ports. Valid bit values specified at each signal port are **0** or **1**.

DESCRIPTION

The **set_bsd_compliance** command specifies the compliance-enable ports and patterns that enable the IEEE 1149.1 boundary-scan functionality for a design.

The functionality of the IEEE 1149.1 boundary-scan logic in certain designs depends on the logic values at the compliance-enable ports for the designs. Compliance-enable ports are defined in ANSI/IEEE Std. 1149.1, section 3-8.

The **set_bsd_compliance** command is mandatory for all IEEE 1149.1 designs that contain at least one compliance-enable port.

EXAMPLES

The following example specifies one compliance-enable pattern named *p1* for the current design:

```
set_bsd_compliance
```

```
prompt> set_bsd_compliance -name p1 \  
-pattern [get_ports c_en*, 1, ntest, 0]
```

SEE ALSO

reset_bsd_configuration(2)

set_bsd_configuration

Specifies the boundary-scan configuration for a design.

SYNTAX

```
status set_bsd_configuration
  [-asynchronous_reset true | false]
  [-default_package package_name]
  [-instruction_encoding binary | one_hot]
  [-ir_width instruction_register_length]
  [-style synchronous | asynchronous]
  [-check_pad_designs none | all | pad_design_list]
  [-control_cell_max_fanout max_fanout]
  [-ieee1149.1_1993 enable | disable]
  [-std std_version_name_list]
  [-rtl enable | disable]
```

Data Types

<i>package_name</i>	string
<i>instruction_register_length</i>	integer
<i>max_fanout</i>	integer
<i>std_version_name_list</i>	list of strings

ARGUMENTS

-asynchronous_reset true | false

Specifies whether or not the boundary-scan circuitry synthesized by the **insert_dft** command is to have an asynchronous test logic reset port. When set to **true** (the default), an asynchronous test logic reset port is used. when set to **false**, there is no asynchronous test logic reset port.

-default_package *package_name*

Specifies the default package for a boundary-scan design. The *package_name* value must correspond to one of the packages for the design read in using the **read_pin_map** command.

-instruction_encoding binary | one_hot

Specifies the instruction encoding scheme to use to generate the decoding logic for the instruction register and the instruction code assignment.

Setting the value to **binary** (the default) causes the **insert_dft** command to use binary coding for the instruction opcodes. This method uses a minimum number of register bits for the instruction register.

Setting the value to **one_hot** causes the **insert_dft** command to use opcodes that have only one bit set to logic 1 with all other bits set to logic 0. The exception is the **BYPASS** instruction, which has all encoding bits set to logic 1. This encoding method uses more instruction register bits, but it simplifies the decoding logic.

-ir_width *instruction_register_length*

Specifies the length of the instruction register, in bits, for the boundary-scan design. Allowed values are the integers **2** through **63**, inclusive. The default is the minimum value calculated by BSD Compiler to fit the set of instructions specified and encoding method to be used.

-style synchronous | asynchronous

Specifies the style of boundary-scan to implement. Valid values are **asynchronous** and **synchronous**. The default style is **synchronous**.

-check_pad_designs none | all | pad_design_list

Specifies the pad designs to check. When set to **none**, the **preview_dft** command does not validate any pad design.

When set to **all** (the default), the **preview_dft** command validates all pad designs instantiated in the design.

When given a *pad_designs_list*, the **preview_dft** command validates all pad designs specified in the list.

The signal type **port** must be specified in the access list of all pad designs that are to be validated. See the man page for the **set_bsd_pad_design** command for details.

-control_cell_max_fanout *max_fanout*

Specifies the maximum number of fanouts allowed for the control cell when the **-share true** option is specified in the **set_boundary_cell** command.

-ieee1149.1_1993 enable | disable

Specifies the mode in which BSD Compiler performs. This option is disabled by default, so BSD Compiler runs in the IEEE1149.1-2001 standard mode. When set to **enable**, BSD Compiler runs in the old IEEE1149.1-1993 standard mode.

This option is obsolete. Use **-std {ieee1149.1_1993}** to enable IEEE1149.1-1993 standard mode.

-std *std_version_name_list*

Specifies the modes in which BSD Compiler performs. You can specify one or more of the following values: **ieee1149.1_1993**, **ieee1149.1_2001**, and **ieee1149.6_2003**.

- When set to **-std ieee1149.1_1993**, BSD Compiler runs in the old IEEE1149.1-1993 standard mode.
- When set to **-std ieee1149.6_2003**, BSD Compiler runs in the IEEE1149.6-2003 standard mode.
- When set to **-std {ieee1149.1_1993 ieee1149.6_2003}**, BSD Compiler runs in the IEEE1149.1-1993 and IEEE1149.6-2003 standard modes.

By default, BSD Compiler runs in the IEEE1149.1-2001 standard mode.

-rtl enable | disable

Specifies the flow in which BSD Compiler should run. This option is disabled by default, so BSD Compiler runs in the default netlist flow.

When set to **enable**, BSD Compiler runs in the new RTL generation flow and generates RTL output. The **write_bsd_rtl** command should be used to generate RTL output file.

DESCRIPTION

The **set_bsd_configuration** command specifies the boundary-scan configuration for a design. Customize the implementation by selecting the synchronization style, the instruction register length, the reset mechanism, and the port-to-pin mapping.

When the **asynchronous** style is selected, the asynchronous signals coming from the TAP controller cell, `capture_dr` and `update_dr`, are connected to the boundary-scan register cells. The asynchronous version of the TAP controller is used.

When the **synchronous** style is selected, the synchronous signals coming from the TAP controller cell, `capture_en`, `update_en`, and TCK are connected to the boundary-scan register cells. The synchronous version of the TAP controller is used. By default, BSD Compiler generates a synchronous boundary scan design.

By specifying the instruction length, you can control the opcode length of the implemented instruction register. By default, BSD Compiler calculates the minimum length of the instruction register needed to encode both the mandatory instruction and those instructions specified using the **set_bsd_instruction** command. This also depends on the encoding scheme selected with the **-instruction_encoding** option.

To reset the TAP controller, you can hold the TMS signal high for a minimum of five TCK clock cycles. You can also specify the implementation of an asynchronous test reset signal (TRST) with the **-asynchronous_reset true** option. In this case, you must either configure a TRST port with the **set_dft_signal -type trst** command, or you must configure an on-chip power-up reset (PUR) cell with the **set_bsd_power_up_reset** command.

The mapping of logical signals onto the physical pins of a component package is described through a port-to-pin map file. Multiple packages for a design can be read in, using the **read_pin_map** command. The sequence of ports listed in the pin mapping file after the line declaring the package name drives the order of the corresponding cells in the boundary-scan register. The default package name specified with the **set_bsd_configuration -default_package** command appears in the generic parameter statement of the BSDL file produced by the **write_bsd** command.

By specifying the **rtl** option, BSD Compiler can be used to generate RTL boundary scan output for a given design.

Use the **report_test -bsd_configuration** command to display the current default package for the design.

Use the **reset_bsd_configuration** command to remove the settings for the design.

EXAMPLES

The following example enables ieee1149.1_1993:

```
prompt> set_bsd_configuration -std ieee1149.1_1993
```

The following example specifies a bypass model for BSR cell insertion. It also specifies that a buffer chain should be inserted for the `shift_dr`, `clock_dr`, `update_dr` and mode control signals, placed every 10 BSR cells:

```
prompt> set_bsd_configuration -cross_hierarchy { u_CORE/u_model } \  
-bsr_buffer_fanout 10
```

The following example enables the RTL generation flow:

```
prompt> set_bsd_configuration -rtl enable
```

SEE ALSO


```
read_pin_map(2)
remove_pin_map(2)
report_bsd_configuration(2)
reset_bsd_configuration(2)
set_bsd_instruction(2)
write_bsd_rtl(2)
```

set_bsd_instruction

Specifies boundary-scan instructions used by the **insert_dft** command for the current design or used by the **check_bsd** command in the verification flow.

SYNTAX

```
status set_bsd_instruction
  [-view existing_dft | spec]
  instruction_list
  [-code inst_code_list]
  [-register register_name]
  [-input_clock_condition clock_conditioning]
  [-output_condition BSR | HIGHZ | NONE]
  [-internal_scan pin_name]
  [-capture_value capture_value_list]
  [-private]
  [-clock_cycles clock_cycle_list]
  [-signature pattern]
  [-high pin_name_list]
  [-low pin_name_list]
  [-sequential_high pin_name_list]
  [-sequential_low pin_name_list]
  [-time real_time]
  [-excluded_bsr_condition CLAMP | NONE]
  [-bsd_init_data init_value]
  [-length register_length]
```

Data Types

<i>instruction_list</i>	list
<i>inst_code_list</i>	list
<i>register_name</i>	string
<i>clock_conditioning</i>	string
<i>pin_name</i>	string
<i>capture_value_list</i>	list
<i>clock_cycle_list</i>	list
<i>pattern</i>	string
<i>pin_name_list</i>	list
<i>real_time</i>	float

ARGUMENTS

-view existing_dft | spec

Specifies the view to which the specification applies. Valid views are **existing_dft** and **spec**.

Set the view to **existing_dft** to refer to the existing instructions in the design. Use the **existing_dft** value when working with BSD-inserted designs. For example, use the following command to specify the instruction for the tool to implement:

```
prompt> set_bsd_instruction bypass -view existing_dft \
      -register BYPASS -code 1111
```

Set the view to **spec**, the default, to refer to instructions that the tool must use when running **check_bsd**. Use this view when boundary-scan synthesis is not performed by **insert_dft** (in the verification flow). For example, to direct the tool that the *my_instr* instruction must be implemented as targeted for the register BOUNDARY, use the following command:

```
prompt> set_bsd_instruction my_instr -view spec \
      -register BYPASS -code 1001
```

instruction_list

Specifies a set of boundary instructions.

-code inst_code_list

Specifies the list of binary codes corresponding to the instructions specified by *instruction_list*. A single instruction can be associated with an opcode. This means that the *instruction_list* must contain a single identifier if **-code inst_code_list** is used. The opcode must have a length equal to the number set by **set_bsd_configuration -ir_width**. By default, the opcode is assigned automatically by BSD Compiler.

When using this option with the **check_bsd** command in the verification flow, opcodes should be specified. Only BYPASS and EXTEST reserved opcodes are inferred automatically.

-register register_name

Selects a standard data register or a user-defined register, previously declared with the **set_dft_signal** and the **set_scan_path** commands, to be connected for serial access between TDI and TDO when the instruction is active. Valid standard data registers are **BOUNDARY** and **BYPASS**. The **BOUNDARY** value is automatically selected for EXTEST, SAMPLE, PRELOAD and INTEST. The **BYPASS** value is automatically selected for BYPASS, CLAMP and HIGHZ. User-defined instructions must specify a register. You cannot specify **BOUNDARY** as a data register for the BYPASS, HIGHZ, CLAMP, IDOCODE, and USERCODE standard instructions. Similarly, **BYPASS** cannot be specified as the data register for the EXTEST, INTEST, SAMPLE_PRELOAD, SAMPLE, PRELOAD, IDCODE, and USERCODE standard instructions.

-input_clock_condition clock_conditioning

Specifies the value that drives the clock signal going into the system when the instruction is active. If you are clocking the TDR with the system clock, set the value to *PI*. If you are using TCK clock, set the value to *TCK*. Valid values are **PI** or **TCK**. The default is **PI**.

-output_condition BSR | HIGHZ | NONE

Specifies how the system outputs are driven when the instruction is active. Valid conditions are as follows:

- **BSR** puts the boundary scan register into EXTEST mode.
- **HIGHZ** puts the output pins into high impedance mode.
- **NONE** puts the boundary scan register into transparent mode (the default).

-internal_scan pin_name

Specifies a hierarchical pin in the design to which the TAP controller's shift-dr signal gated with the decoded instruction is connected.

-capture_value capture_value_list

Defines the capture value of the DEVICE-ID register for the IDCODE instruction. The capture value contains 32 digits. Ensure that the least significant bit (LSB) of the capture value is set to 1 for the IDCODE capture value.

-private

Specifies that the instructions are considered private. Private instructions have the **INSTRUCTION_PRIVATE** attribute in the BSDL file generated by the tool. The data registers of private instructions are not shown in the BSDL file. No test patterns are generated for private instructions. By default, the instructions are not considered private.

-clock_cycles *clock_cycle_list*

Lists the clock port and integer pairs that specify the minimum number of clock cycles on the clock port required for the design to stay in the Run-Test/Idle TAP controller state to ensure completion of the INTEST or the RUNBIST instruction.

-signature *pattern*

Specifies the state of the boundary-scan register after execution of the RUNBIST instruction. The *pattern* argument correlates to the <det pattern> argument defined in section B.8.15 of the Supplement to IEEE Std. 1149.1.

-high *pin_name_list*

Specifies a list of pins to be asserted high when the specified instructions are selected. The *pin_name_list* contains the hierarchical names of the pins. There is no default for this option. This option can be used with any instruction, including standard instructions.

-low *pin_name_list*

Specifies a list of pins to be asserted low when the specified instructions are selected. The *pin_name_list* contains the hierarchical names of the pins. There is no default for this option. This option can be used with any instruction, including standard instructions.

-sequential_high *pin_name_list*

Specifies a list of pins to be glitchlessly asserted high when the specified instructions are selected. This option is similar to the **-high** option, except that the combinational decode logic is registered to prevent glitches at the specified pins.

-sequential_low *pin_name_list*

Specifies a list of pins to be glitchlessly asserted low when the specified instructions are selected. This option is similar to the **-low** option, except that the combinational decode logic is registered to prevent glitches at the specified pins.

-time *real_time*

Specifies a real number that indicates the time duration in nanoseconds for the EXTEST_TRAIN and EXTEST_PULSE instructions in the Run-Test-Idle (RTI) state, as well as for INTEST and RUNBIST. The tool returns an error if this option is used with other instructions.

-excluded_bsr_condition CLAMP | NONE

Specifies the condition of BSR cells excluded from the specified short BSR chain. The following values are valid for this option:

- **CLAMP** indicates that all BSR cells that are not part of the specified BSR chain are conditioned as if CLAMP instruction is active.
- **NONE** indicates that all BSR cells that are not part of the specified BSR chain are kept transparent while the user instruction is active.

Note: The following error occurs when a value other than CLAMP or NONE is used with this option:

Invalid value '%s' specified for option '-excluded_bsr_condition'.

-bsd_init_data *init_value*

Specifies an optional initialization value to be loaded into the test data register associated with this instruction at the beginning of the test program created by the **create_bsd_patterns -setup_instructions** command.

By default, no initialization value is loaded at the beginning of the test program.

DESCRIPTION

The **set_bsd_instruction** command specifies one or more boundary-scan instructions that the **insert_dft** command implements for the current design. The **check_bsd** command infers these instructions in the verification flow when the **-infer_instructions** switch is set to **true** or if the **insert_dft** command does not perform boundary-scan synthesis. Alternatively, all boundary-scan synthesis decoded opcodes need to be listed. The default value for the **-infer_instructions** switch is **false**, in which case, it checks compliance of the instructions specified in the run script. You can specify a list of IEEE Std 1149.1 instructions. The keywords for the standard instructions are BYPASS, EXTEST, SAMPLE, PRELOAD, IDCODE, CLAMP, HIGHZ, and USERCODE.

For all other standard and private instructions, including INTEST and RUNBIST, you can specify their corresponding binary code, input clock conditioning, user-defined register, and output conditioning.

If you do not specify any binary code, the **insert_dft** command selects one automatically in the design flow. However, you must specify opcodes in the Verification flow for the BYPASS, SAMPLE, PRELOAD, and EXTEST instructions when the **set_bsd_instruction** command is used with the **check_bsd** command and the **-infer_instructions** switch is set to **true**. The input clock conditioning and output conditioning (BSR or HIGHZ) specifications are required for the standard instructions INTEST and RUNBIST.

By default, the capture and update clock pins of the Test Data Register (TDR) specified with instructions are not gated, that is, the pins are always driven by TCK. To gate the clock pins of a TDR with a MUX so that the pins are driven by TCK when the instruction that selects the TDR is active and driven by system logic when the instruction is not active, set the following variable to **true** prior to BSD insertion:

```
prompt> set test_bsd_synthesis_gated_tck true
```

You can use the **-high**, **-low**, **-sequential_high**, and **-sequential_low** options to specify one or more design pins to be asserted high or low when an instruction in the *instruction_list* is loaded. The **-high** and **-low** options create a combinational decode of the instruction register Q pins, which provides a minimal-area decode but can produce glitches in the assertion signal. If you are asserting glitch-sensitive design pins such as asynchronous set/reset pins, use the **-sequential_high** and **-sequential_low** options instead. These options create a registered decode of the instruction register D pins, which increases sequential area but produces a glitch-free assertion signal.

Note that registered decodes cannot be mixed with combinational decodes. If you specify the **-sequential_high** or **-sequential_low** option for any pin, then all **-high** and **-low** pin assertions (for all instructions) also become registered.

The **-clock_cycle** option specifies a list of clock port and integer pairs, and each pair specifies the minimum number of clock cycles on the specified clock port that the design needs to stay in the Run-Test or the Idle TAP controller state to ensure completion of the RUNBIST, INTEST, EXTEST_PULSE, and EXTEST_TRAIN instructions.

The **-time** option applies to EXTEST_PULSE and EXTEST instructions. For the EXTEST_PULSE instruction, the *real_time* value specifies the minimum wait time in nanoseconds in the Run-Test or the Idle TAP controller state. Note that you can specify only one of the **-clock_cycles** or **-time** options for this instruction.

For the EXTEST_TRAIN instruction, the *real_time* value specifies the maximum time in nanoseconds prior to the exit of the Run-Test or the Idle TAP controller state within which the specified minimum number of pulses specified with the **-clock_cycles** option should occur for EXTEST_TRAIN instruction. Note that the **-time** option must always be used along with **-clock_cycles** for this instruction.

The **-signature** option specifies a binary string that represents the signature produced by the RUNBIST instruction.

The **-capture_value** option allows you to specify a 32-bit binary or hexadecimal value and pins in the design, that provide the capture value, to be connected to the DIR register. This option also allows you to specify a mix of bits and design pins.

Specify a valid user code value for the USERCODE instruction. The user code value can only be specified for the USERCODE instruction. Specify binary or hexadecimal bit-patterns using the size-constant Verilog syntax as shown in the following example:

```
<number of binary bits>'[b|h|B|H]<binary or hex value>
```

Specify pin names with the full hierarchical pin name. Specify a bus representing a collection of pins using the following syntax:

```
<name of the bus>[<upper bit>:< lower bit>]
```

To review your boundary-scan instruction specifications, use the **preview_dft -bsd all** command. To implement the boundary-scan instruction set, use the **insert_dft** command. To delete boundary-scan instruction specifications, use the **remove_bsd_instruction** command.

EXAMPLES

The following example directs **insert_dft** to implement the HIGHZ and CLAMP instructions as a part of the synthesized boundary-scan circuitry:

```
prompt> set_bsd_instruction -view spec {HIGHZ CLAMP}
```

The following example directs **insert_dft** to implement the user-defined instruction named *UDI1*, whose binary code is 1010, and to select the boundary-scan register to be connected for serial access between TDI and TDO. This puts the outputs in high-impedance instructions as a part of the synthesized boundary-scan circuitry.

```
prompt> set_bsd_instruction -view spec UDI1 -register BOUNDARY \
      -code {1010} -output_conditioning HIGHZ
```

The following example directs **insert_dft** to implement the INTEST instructions, and to drive the system clock by **TCK**:

```
prompt> set_bsd_instruction -view spec INTEST \
      -input_clock_condition TCK
```

The following example directs **insert_dft** to implement the USERCODE instruction:

```
prompt> set_bsd_instruction -view spec USERCODE -code {0110} \
      -user_code_val {version_reg/v3 version_reg/v2 \
      version_reg/v1 version_reg/v0 16'b1111111110000000 12'h2ab}
```

SEE ALSO

```
check_bsd(2)
insert_dft(2)
preview_dft(2)
remove_bsd_instruction(2)
set_bsd_configuration(2)
set_scan_path(2)
```

set_bsd_linkage_port

Identifies the linkage ports in your design.

SYNTAX

```
status set_bsd_linkage_port  
-port_list list_of_ports
```

Data Types

list_of_ports list

ARGUMENTS

-port_list *list_of_ports*

Specifies the ports of the current design to be considered as linkage ports.

DESCRIPTION

The **set_bsd_linkage_port** command identifies the ports of the current design to be considered as linkage ports. Such ports may be analog ports, power or ground ports, or any port driven by a black-box cell that you do not want to consider for boundary-scan insertion.

The following actions are not performed for ports declared as linkage ports by the **set_bsd_linkage_port** command:

- Checking for pad cells connected to the port during boundary-scan insertion.
- Insertion of a boundary-scan cell to be connected to the port.
- Compliance checking on the port.
- Toggling of the port in test vector generation.

The linkage ports are listed as linkage bits in the BSDL.

To preview the boundary-scan implementation, use the **preview_dft** command with the **-bsd all** option. To implement the boundary-scan logic, use the **insert_dft** command. To delete boundary-scan linkage port specifications, use the **remove_bsd_specification** command with the **-linkage_port** option.

EXAMPLES

The following example shows how the PLL_1, PLL_2, and PLL_3 ports are identified as linkage bits:

```
prompt> set_bsd_linkage_port -port_list {PLL_1, PLL_2, PLL_3}
```

SEE ALSO

insert_dft(2)
preview_dft(2)

set_bsd_power_up_reset

Specifies and characterizes the power-up reset cell for the current design.

SYNTAX

```
status set_bsd_power_up_reset  
-cell_name cell_name  
-reset_pin_name reset_pin_name  
-active high | low  
[-delay power_up_reset_delay]
```

Data Types

```
cell_name           string  
reset_pin_name      string  
power_up_reset_delay integer
```

ARGUMENTS

-cell_name *cell_name*

Specifies the instance name of the power-up reset cell.

-reset_pin_name *reset_pin_name*

Specifies the pin name of the power-up reset cell which generates a reset pulse upon power-on.

-active high | low

Specifies whether the power-up reset pulse is active **high** or active **low**.

-delay *power_up_reset_delay*

Specifies the initial power-up reset delay, which is measured from the time power is switched on to the time the TAP controller resets to the test-logic-reset state, in library time units.

DESCRIPTION

This command specifies and characterizes the power-up reset cell instance for the design. You must use the **-delay** option with the *power_up_reset_delay* value if the design does not have the (optional) TRST test access port.

EXAMPLES

The following example shows how to specify the *PUR* instance as the power-up reset with a pin named *reset_out* as the pin at which an active-high reset pulse is generated upon power-up with a delay of *1300* library time units:

```
prompt> set_bsd_power_up_reset -cell_name PUR \  
       -reset_pin_name reset_out -active high -delay 1300
```

SEE ALSO

- check_bsd(2)
- create_bsd_patterns(2)
- insert_dft(2)
- preview_dft(2)
- set_bsd_configuration(2)
- write_test(2)
- test_default_period(3)

set_case_analysis

Specifies that a port or pin is at a constant 1 or 0 logic value or has only a rising or only a falling transition applied to it.

SYNTAX

```
status set_case_analysis  
  value  
  port_or_pin_list
```

Data Types

```
value          integer or string  
port_or_pin_list list
```

ARGUMENTS

value

Specifies the constant logic value or the transition assigned to the specified port or pin. The valid constant values are **0**, **1**, **zero**, and **one**. The valid transition values are **rising**, **falling**, **rise**, and **fall**.

port_or_pin_list

Lists the ports or pins to which the *value* applies.

DESCRIPTION

This command specifies that a port or pin is at a constant 1 or 0 logic value or has only a rising or only a falling transition applied to it.

Case analysis is a way to specify a given mode of the design without altering the netlist structure. For the current timing analysis session, you can specify that certain signals are at a constant value (1 or 0) or that only one type of transition (rising or falling) is to be applied. When you specify case analysis as a constant value, that value is propagated forward through the network as long as a controlling value for the traversed logic is at the constant value.

For example, if you specify that one of the inputs of a NAND gate is a constant logic 0, that condition is propagated to the NAND output, which is then considered a constant logic 1 value. This constant value is in turn propagated to the inputs of all cells driven by this signal, and so on.

In the event of case analysis set as a transition (such as rising), the specified pin or port is only considered for timing analysis with that type of transition. The other transition (such as falling) is not considered.

You can use case analysis (in addition to the mode commands) to fully specify the operating mode of a design. For example, you can use the **set_mode** command to specify a design that instantiates models with a TESTMODE signal that is disabled during timing analysis. In addition, if a TESTMODE signal exists in the design, you can set it to a constant logic value so that all test logic that the TESTMODE signal controls is disabled.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example specifies that a port named IN1 is at a constant 0 logic value.

```
prompt> set_case_analysis 0 IN1
```

The following example specifies that the U1/U2/A pin is considered only for a rising transition. The falling transition on this pin is not considered.

```
prompt> set_case_analysis rising U1/U2/A
```

SEE ALSO

- `remove_case_analysis(2)`
- `report_case_analysis(2)`
- `set_mode(2)`
- `reset_mode(2)`

set_cell_degradation

Sets the **cell_degradation** attribute to a specified value on the specified ports or designs.

SYNTAX

```
status set_cell_degradation  
      cell_degradation_value  
      object_list
```

Data Types

<i>cell_degradation_value</i>	float
<i>object_list</i>	list

ARGUMENTS

cell_degradation_value

Specifies a capacitance value for setting the **cell_degradation** attribute. You must enter the *cell_degradation_value* argument in capacitance units consistent with those used by the technology library during optimization. For example, if the library specifies capacitance values in picofarads, you must express the *cell_degradation_value* argument in picofarads.

object_list

Specifies the input ports on which to set the **cell_degradation** attribute.

DESCRIPTION

This command sets the **cell_degradation** attribute on the specified ports or designs to the value specified in the *cell_degradation_value* argument.

During optimization, the tool attempts to ensure that the capacitance value for a net is less than the *cell_degradation_value* if the value of the **compile_fix_cell_degradation** variable is **true**.

If cell degradation tables are specified in the technology library (implicit constraints), the tool automatically tries to meet them if the value of the **compile_fix_cell_degradation** variable is **true**. The cell degradation tables give the maximum capacitance that can be driven by a cell as a function of the transition times at the inputs of the cell.

By default, a port has no cell degradation constraint.

The **cell_degradation** attribute and the **max_capacitance**, **max_fanout**, and **max_transition** attributes are design rule constraints; the **max_delay** and **max_area** attributes are optimization constraints. Design rule constraints reflect those technology-specific

restrictions that must be met for a design to function correctly. Optimization constraints reflect desirable, but not crucial, goals and restrictions for the operation of a design. The tool attempts to meet all constraints placed on a design, but gives priority to design rule constraints in the optimization process. Therefore, optimization gives preference to **cell_degradation** and other design rule constraints, even if they adversely affect the optimization constraints on a design.

To get information about optimization and design rule constraints, use the **report_constraint** command. To remove the **cell_degradation** attribute from a port, use the **remove_attribute** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example sets a maximum capacitance value of 2.0 units on the port named `late_riser`:

```
prompt> set_cell_degradation 2.0 late_riser
```

SEE ALSO

- `all_inputs(2)`
- `all_outputs(2)`
- `characterize(2)`
- `compile(2)`
- `remove_attribute(2)`
- `report_constraint(2)`
- `set_max_capacitance(2)`
- `compile_fix_cell_degradation(3)`

set_cell_internal_power

Sets or removes the **power_value** attribute on the specified pins. The value represents the power consumption for a single toggle of each pin.

SYNTAX

```
status set_cell_internal_power
      pin_list [power_value [unit]]
      | -delete_all
```

Data Types

```
pin_list    collection
power_value float
unit        string
```

ARGUMENTS

pin_list

Specifies the pins on which to either set the **power_value** attribute or remove previously-annotated **power_value** attributes.

This option cannot be used with the **-delete_all** option.

power_value

Specifies the value with which to set the **power_value** attribute on the specified pins. This value represents the power consumption for a single toggle of the pin.

If you do not specify the *power_value* argument, any existing **power_value** attributes are removed from the specified pins.

This option cannot be used with the **-delete_all** option.

unit

Specifies the unit of the power value. Allowed values for *unit* are as follows:

```
GW  MW  KW  W  mW
uW  nW  pW  fW  aW
```

If you do not specify the *unit* argument, the tool uses the library power unit. If the library does not have any units, the tool displays an error message.

This option cannot be used with the **-delete_all** option.

-delete_all

Deletes all of the internal power annotations from all of the pins in the design.

This option cannot be used with the *pin_list*, *power_value*, or *unit* arguments.

DESCRIPTION

This command sets or removes the **power_value** attribute on the specified pins.

If a cell has at least one pin with a **power_value** attribute, its internal power is calculated as the sum of the pin power values, where each pin power value is calculated by multiplying the annotated power value on that pin by the pin toggle rate.

You can use this command to override a cell's library power characterization in situations where that characterization does not apply. A common use is when you manually replace an entire cloud of logic with a single cell and you want the single cell's power consumption to represent that of the cloud of logic. For example, if you replace a clock tree by a single buffer cell, you can set the **power_value** attribute on the output pin of the buffer cell with the value of the power consumption for one clock toggle of the entire clock tree. Although the buffer cell might have been power-characterized in the library, its power consumption is now calculated by using the value of the **power_value** attribute set by the **set_cell_internal_power** command.

Do not use this command as a routine part of the power characterization flow. It is considered a best practice to use a cell's library power characterization unless there is a reason to override it, as described above.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example shows how an instance-specific cell is power-characterized by using the **set_cell_internal_power** command:

```
prompt> set_cell_internal_power [get_pins U123/A] 1.234 uW
```

```
prompt> set_cell_internal_power [get_pins U123/B] 1.567 uW
```

```
prompt> set_cell_internal_power [get_pins U123/*] -1.000 nW
```

```
prompt> set_cell_internal_power -delete_all
```

SEE ALSO

`report_power(2)`

set_cell_location

Specifies the physical location, orientation, and dont_touch status for leaf cells. This command is supported in Design Compiler topographical mode only.

SYNTAX

```
status set_cell_location
  -coordinates {x y}
  [-orientation target_orient]
  [-fixed]
  object_list
```

Data Types

<i>x</i>	float
<i>y</i>	float
<i>target_orient</i>	string
<i>object_list</i>	collection

ARGUMENTS

-coordinates {*x y*}

Specifies the lower-left coordinates of the specified cells. The numbers are in microns relative to the chip origin.

-orientation *target_orient*

Specifies the target orientation.

Valid values are

N or **0**: Nominal orientation (north)

S or **180**: 180-degree rotation (south)

E or **270**: 270-degree counterclockwise rotation (east)

W or **90**: 90-degree counterclockwise rotation (west)

FN or **0-mirror**: reflection in the y-axis (flipped north)

FS or **180-mirror**: 180-degree rotation followed by reflection in the y-axis (flipped south)

FE or **90-mirror**: 90-degree counterclockwise rotation followed by reflection in the y-axis (flipped east)

FW or **270-mirror**: 270-degree counterclockwise rotation followed by reflection in the y-axis (flipped west)

-fixed

Sets the **dont_touch** attribute on the specified cells to prevent modification or replacement of these cells during optimization.

object_list

Specifies the leaf cells on which to set the physical location, orientation, and dont_touch status for leaf cells. Typically, this command uses a single leaf cell as the argument.

DESCRIPTION

The **set_cell_location** command specifies the physical location, orientation, and dont_touch status for leaf cells.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the location of the lower-left corner of the cell named INST_1 to (100 100):

```
prompt> set_cell_location -coordinates { 100 100 } INST_1
```

SEE ALSO

report_physical_constraints(2)
write_physical_constraints(2)

set_cell_mode

Selects the mode of a component.

SYNTAX

```
status set_cell_mode  
  [mode_list]  
  [instance_list]
```

Data Types

```
mode_list    list  
instance_list list
```

ARGUMENTS

mode_list

Specifies the active modes for timing analysis. All other modes of the same mode group are implicitly inactive. The specified modes are active on the given instance list. If multiple modes are selected they must be in separate mode groups.

instance_list

Specifies the cells for which the specified modes are active.

DESCRIPTION

The **set_cell_mode** command is used to select the active mode of a cell that has several functional modes. The other modes in the same mode group that are not selected become disabled.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following command selects the READ mode for the RAM instance Uram1.

```
prompt> set_cell_mode READ Uram1
```

The following example applies the **set_cell_mode** command to a cell that has two mode groups, rw0 and rw1, and then shows the output of **report_cell_mode**. In this example mode group rw0 is set to mode read0, and mode group rw1 is set to mode write1.

```
prompt> set_cell_mode [list read0 write1] U11/core
```

```
prompt> report_cell_mode
```

```
*****  
Report : mode  
Design : top_ram2  
Version: 2013.03-ICC-SP  
Date   : Wed Feb 13 14:12:56 2013  
*****
```

Cell	Mode(Group)	Status

U11/core	read0 (rw0)	ENABLED
	write0 (rw0)	DISABLED
	read1 (rw1)	DISABLED
	write1 (rw1)	ENABLED

SEE ALSO

report_cell_mode(2)
reset_cell_mode(2)

set_check_library_options

Sets options for the **check_library** command for various logic library and physical library checks.

SYNTAX

```
status set_check_library_options
  [-cell_area]
  [-cell_footprint]
  [-bus_delimiter]
  [-tech_consistency]
  [-view_comparison]
  [-same_name_cell]
  [-signal_em]
  [-antenna]
  [-rectilinear_cell]
  [-physical_only_cell]
  [-phys_property {property_list}]
  [-rail_view_data]
  [-routeability]
  [-routability]
  [-min_pin_layer layer_name]
  [-tech]
  [-placement]
  [-pattern_must_join_pin]
  [-pattern_must_join_pin_exclusion_list lib_pin_list]
  [-drc]
  [-scaling {scaling_types}]
  [-mcmm]
  [-upf]
  [-optimization {power}]
  [-compare {construct | attribute | value}]
  [-tolerance {type relative_tolerance absolute_tolerance}]
  [-validate {timing noise [value]}]
  [-analyze {nominal_vs_sigma | table_trend | table_bound | table_slope | table_index | sensitivity | voltage_range}]
  [-criteria {criteria_spec}]
  [-group_attribute {groups_or_attributes}]
  [-report_format {format_spec}]
  [-leq]
  [-by_group_order]
  [-char_integrity]
  [-significant_digits digits]
  [-physical]
  [-logic_vs_physical]
  [-logic]
  [-reset]
  [-all]
```

Data Types

<i>property_list</i>	list
<i>scaling_types</i>	list
<i>type</i>	string
<i>relative_tolerance</i>	float
<i>absolute_tolerance</i>	float
<i>criteria_spec</i>	list
<i>groups_or_attributes</i>	list
<i>format_spec</i>	list

ARGUMENTS

-cell_area

Compares the cell area defined by the area attribute in the logic library with the area determined by the cell place-and-route boundary in the physical library.

In a FRAM view, the rectangular area covered by the place-and-route boundary for a standard cell, or the area enclosed by the cell boundary of a macro in a polygon, is compared with the cell area in the .db file. This comparison determines an area ratio for each cell in the .db file. If the ratio is found to be the same for all the cells in the logic library, the cell areas are considered consistent. However, if the ratio for a cell deviates from the normal or average ratio for this library by a margin of 5 percent or more, the cell areas are considered inconsistent.

There is no area check for the pad cells because they are special cells located at the chip boundary, and are not used as internal gates. They are considered to have zero area, and are therefore not checked for area.

This option also enables the default logic-versus-physical checks.

-cell_footprint

Checks that the cell place-and-route boundary in the physical library is consistent among a set of cells when the **cell_footprint** attribute is specified in the logic library. This option reports cell names and their place-and-route boundaries.

This option also enables default logic versus physical checks.

-bus_delimiter

Reports bus delimiters in the logic and physical libraries. If the library does not contain a bus delimiter, the field remains blank in the report.

This option also enables default logic versus physical checks.

-tech_consistency

Checks for consistency of technology data between the main library or design library and each associated reference library. The option checks the physical libraries for missing layer data and mismatched technology data. This option is not supported in NDM mode.

-view_comparison

Checks for existence of the CEL and FRAM views in the library, and for mismatched time stamps between these views. In the report table for missing views, the CEL and FRAM columns list the cell name and version number. The missing views are marked by an X.

In the report table for the mismatched views, the CEL version and FRAM version columns list the version numbers. The CEL and

FRAM "modified time" column shows the time when each view was last modified. If the FRAM view's "modified time" is earlier than that of the CEL view, it is marked as mismatched.

The internal view creation or modification time in the Milkyway database is checked. No check is performed on the cell content. If the FRAM views were read from a library exchange format (LEF) database, and CEL views were streamed in later, the views are reported as mismatched. You can ignore the report in such cases. This option is not supported in NDM mode.

-same_name_cell

Checks for cells with identical names among the specified main or design library and all physical reference libraries linked to the specified library. If there are cells with the same name in multiple reference libraries, the first cell in the reference control file is used. The remaining cells are ignored. This option does not check naming among the logic libraries.

-signal_em

Checks the signal electromigration rule for the current model and the model type in each routing layer.

-antenna

Checks for missing antenna properties of cells and antenna rules in the layers of the specified main library.

An antenna property report table lists the names of the cells and pins that are missing an antenna property and shows the missing property type. For example, the following conditions are reported:

- * An input pin is missing the gate size.
- * An output pin is missing diode protection information.
- * A macro is missing the hierarchical antenna property.

This option reports the mode, diode mode, default metal ratio, default cut ratio, and maximum ratio for each antenna rule.

-rectilinear_cell

Checks and reports the cell names, types, and coordinates of cells with rectilinear boundaries. This option is not supported in NDM mode.

-physical_only_cell

Checks and reports the cell names, types, and properties of physical-only cells. Physical-only cells are corner pad cells, tap cells, chip cells, I/O pad cells, cover cells, cells that only have power and ground pins, and filler cells with or without metal. This option is not supported in NDM mode.

-phys_property {*property_list*}

Specifies a list of physical properties to check. You can specify one or more of the following values: **place**, **route**, **cell**, or **metal_density**.

- **place** reports the following placement properties of each standard cell:
 - * Place-and-route boundary as represented by its lower-left and upper-right coordinates.
 - * Cell height relative to the unit-tile height, such as 2xH for a cell with twice the tile height.
 - * Coordinate, the bottom-left location of a cell with unit-tile height, or for a multiple-height cell, the left locations at each unit-tile height.
 - * Tile pattern representing possible cell orientations that have combinations of R0, R90, R160, R0_MX, R0_MY, 90_MX, and R90_MY.
 - * Remarks reporting the place-and-route boundary mismatch with the tile pattern. If there is a mismatch, use the **cmSetMultiHeightProperty** command in the Milkyway Environment tool to set the tile patterns of multiple-height cells.

* Macros report the cell boundary and height.

When you specify the **place** option, it also lists the main library and its reference library names with paths, if any, and the unit tiles, names, and sizes. The unit-tile size is reported in the width x height format. If the library does not have a unit-tile, the unit-tile size is not reported. In this case, the unit-tile of the main library is used in the design.

- **route** checks and reports the routing properties of each routing layer, including the preferred direction, track direction, offset, pitch, and remarks. The remarks column is marked "OK" for zero or half-pitch offset. Otherwise, it shows "pitch!=0" or "offset!=0.5*pitch".
- **cell** reports the cell and pin properties for each cell, including the pin name, direction, and type. It also reports the number of cells with multiple power and ground pins.
- **metal_density** checks the macro metal density data for the cells specified by the **check_library** command with the **-cells** option. The option checks if a cell has at least one of the following errors:

* The metal density data in the CEL view and the FRAM view are inconsistent.

* The metal density windows do not cover the entire area on a layer.

The option lists all the cells with metal density errors in a table with columns for the cell name, data inconsistency between the CEL view and the FRAM view, and the layer where the density windows do not cover the entire cell area.

This option is not supported in NDM mode.

-rail view_data

Checks the rail data consistency between the FRAM and CONN views, and between the FRAM and PARA views. The view_data includes FRAM, CONN and PARA views. If the PG port type, net type, or number does not match in the CONN and FRAM views, the **check_library** command generates a warning message. This option is not supported in NDM mode.

-routeability

-routability

Checks the physical pin on-track accessibility and quality of the defined wire tracks. The option reports the total number of pins without optimal on-track routability and lists the pin names, directions, layers, and tracks for each cell with this issue.

In the track column, an H denotes that the pin is not accessible on a horizontal track, a V denotes that the pin is not accessible on a vertical track, and H&V denotes that the pin is not accessible on both horizontal and vertical tracks.

If a pin is reported as not accessible on-track, the pin might be routed by an off-track wire during detail routing. However, if too many pins do not have on-track routability, adjust the offset values of the wire track preferably to zero or half-pitch. Rerun the **create_lib_track** command to reduce the number of pins that lack optimal accessibility. For NDM mode, use -routability.

-min_pin_layer layer_name

Specifies the minimum layer name. Pins on layers lower than this layer are checked for existence. This option goes with **-routability**.

-tech

Checks for technology data consistency within a single specified library. This is different from the **-tech_consistency** option, which checks the consistency between different libraries. The messages resulting from using the **-tech** option are similar to those generated during library creation or technology data replacement.

This option requires write permission for the directory where the library resides. If the directory does not have the write permission, the command copies the library to a local directory and checks the copy there.

-placement

Checks cell placement constraint, layers, and sites. This option is supported only in NDM mode.

-pattern_must_join_pin

Checks pattern must-join related items. This option goes with **-routability**. This option is supported only in NDM mode.

-pattern_must_join_pin_exclusion_list lib_pin_list

Specifies a collection of lib_pins that don't need the pattern must-join check. This option goes with **-pattern_must_join_pin**.

-drc

Enables DRC checking of the FRAM routing views in the specified library. It checks the following design rules:

```
wire minWidth
minEdgeLength
minEnclosedArea
minSpacing
cutWidth
cutSpacing
minEnclosure
sameNetMinSpacing
maxStackLevel
fatTbISpacing
metal-via spacing (minSpacing in DesignRule section of technology file)
```

This option lists the cells with DRC violations in a table with columns for showing the cell name, type, and error cell. Check the error cells for detailed information about the DRC violations. This option requires write permission for the directory where the library resides. This option is hidden in NDM mode.

-scaling {scaling_types}

Specifies a list of logic library consistency checks for various types of scaling.

You can specify one or more of the following strings:

- **timing** specifies logic library consistency checking for composite current source (CCS) timing scaling.
- **noise** specifies logic library consistency checking for CCS noise scaling.
- **power** specifies logic library consistency checking for CCS or nonlinear delay model (NLDM) power scaling, including driver and receiver models, load indexes, base curve_x, waveform, noise, and power models.

-mcmm

Specifies logic library consistency checking for multicorners-multimode, including the power_down_function attribute and power data.

-upf

Specifies logic library consistency checking for multivoltage and IEEE 1801, also known as Unified Power Format (UPF), and includes power special cells, the pg_pin, voltage_names, and power_down_function attributes, and power data.

-optimization {power}

Checks the libraries for various types of optimization. The **power** option checks for power optimization related issues in the libraries, namely, missing high threshold_voltage_group cells specified by the **target_library** variable or **check_library - logic_library_name** option (the latter having higher priority). When you use the **-optimization** option, use the **-criteria** option to specify the low threshold_voltage_group names. With optimization power checking, the tool checks for the presence of high-threshold-voltage library cells having the same logic functions as the low-threshold-voltage library cells.

-compare {construct | attribute | value}

Specifies the type or types of information compared between logic libraries, which can be one or more of the keywords **construct**, **attribute**, or **value**. For **attribute** or **value** comparison, only two libraries are compared; if more than two libraries are specified, only the first two are compared.

- **-compare {construct}** checks to see if constructs or attributes are missing.
- **-compare {attribute}** checks to see if constructs and attributes are missing and if the attribute values are inconsistent. This option checks all groups, subgroups, and attribute values except characterization values. It is a superset of the **-compare {construct}** option.
- **-compare {value}** compares values of each group and attribute between the libraries. This option is a superset of the **-compare {attribute}** option.

In comparing characterization values, if the source is not from the .lib library files, the values are not reported, such as vector and capacitance values.

The comparison of the characterization values is controlled by the tolerances specified by the **-tolerance** option, or by the default values if the **-tolerance** option is not used. For minimum and maximum library checking and scaling group library checking, you should use the **-compare {attribute}** option to check all the library contents except for characterization values.

When you specify this option, the following default checks are also performed: missing cells, missing and mismatched pins, and timing arcs.

-tolerance {type relative_tolerance absolute_tolerance}

Specifies a relative tolerance and an absolute tolerance for characterization value comparison, such as delay values. The format is

```
-tolerance {type rel_tol abs_tol}
```

The valid values for the *type* argument are delay, slew, delay_ocv, slew_ocv, delay_ccsn, slew_ccsn, delay_sensitivity, slew_sensitivity, constraint, slew_index, load_index, time, power, current, energy, and capacitance. If the type is not specified, the values are for the output load capacitance. When you specify an absolute tolerance, do not include the unit. The units are fixed and can be reported by the **set_check_library_options** command. For example, specify the tolerance values for delay and slew as shown below:

```
-tolerance {delay 0.1 0.2 slew 0.3 0.4}
```

If you do not specify tolerance values for a type, the default values are used. The default values that are set as follows, in compliance with PrimeTime and the Library Quality Assurance System:

```
Relative tolerance for load index : 0.01
Absolute tolerance for load index : 0.001pF
Relative tolerance for time       : 0.04
Absolute tolerance for time       : 0.015ns
Relative tolerance for power      : 0.04
Absolute tolerance for power      : 5pW
Relative tolerance for delay      : 0.02
Absolute tolerance for delay      : 0.005ns
Relative tolerance for slew       : 0.03
Absolute tolerance for slew       : 0.0075ns
Relative tolerance for ocv delay  : 0.02
Absolute tolerance for ocv delay  : 0.005ns
Relative tolerance for ocv slew   : 0.03
Absolute tolerance for ocv slew   : 0.0075ns
Relative tolerance for ccsn delay : 0.03
Absolute tolerance for ccsn delay : 0.003ns
Relative tolerance for ccsn slew  : 0.05
Absolute tolerance for ccsn slew  : 0.005ns
```

Relative tolerance for sensitivity delay : 0.1
 Absolute tolerance for sensitivity delay : 0.01ns
 Relative tolerance for sensitivity slew : 0.15
 Absolute tolerance for sensitivity slew : 0.015ns

-validate {timing noise value}

Specifies logic library consistency checking for library characterization between different models. The **{timing}** option is for validation between different timing models (NLDM versus CCS). The optional **value** is for validation of the characterization value between models of different dimensions, such as a one-dimensional NLDM and a two-dimensional CCS model. The **{noise}** option is for validation of delay/slew between CCSN and NLDM.

-analyze {nominal_vs_sigma | table_trend | table_bound | table_slope | table_index | sensitivity | voltage_range}

Specifies what is analyzed. You can specify any of the following values:

- **nominal_vs_sigma** analyzes multiple characterization tables. In the case of variation-aware models, analyze nominal, -sigma, and +sigma tables to see the trend and standard deviation for each slew and load index grid of a linear model. The linear model is modeled by the **va_parameters** option. Specify the **va_parameters** in the **-group_attribute** option. If you do not specify the **va_parameters**, the variation-aware models for all **va_parameters** are analyzed. Wildcards apply to **va_parameters**.
- **table_trend** analyzes a characterization table for trend. It analyzes the trend within the table to check if the values change monotonically in terms of slew or load indexes.
- **table_bound** analyzes a characterization table for bounds. It analyzes the values of the table to check if the values exceed the upper or lower bound.
- **table_slope** analyzes a characterization table for slope. It analyzes the slope formed by two adjacent grid points within the table to check if the slope value exceeds the maximum or minimum slope criteria. The slope calculation uses normalized values to eliminate the effect of the unit and table types.
- **table_index** analyzes the table index accuracy by calculating the ratio of each pair of adjacent indexes (next index to current index) and reports the index tables if at least one ratio meets the condition specified by the **-criteria {table_type xfactor>min_ratio}** option.
- **sensitivity** analyzes a cell's sensitivity. It reports the condition that the CCSN model response to the clamped waveform has an output delay or output slew that deviates from the CCSN model response to the original or normal non-clamped version of the waveform.
- **voltage_range** analyzes output waveform's voltage swing to see if it reaches 95% of VDD.

-criteria {criteria_spec}

Specifies analysis criteria for variation analysis (multiple tables), non-variation analysis (single table), or low threshold_voltage_group for power optimization. This option is used with the **-analyze** or **-optimization** option. It is specified in the following format:

```
{std_error>m.m slope>n.n trend=trend_symbol |
 table_type upper_bound=upper_bound_value lower_bound=lower_bound_val |
 table_type[table_sub_index] min_slope=min_value max_slope=max_value |
 table_type xfactor>min_ratio clamping_ratio=ratio_val}
```

The **std_error** and **slope** are used by the linear regression models. The **trend_symbol** includes the following eight symbols:

- / monotonically increasing
- \ monotonically decreasing
- ^ non-monotonically up
- V non-monotonically down
- flat

- M Multiple peaks
- W Multiple troughs
- N ≥ 2 peaks with 2nd not turning low

To specify the monotonically decreasing trend, use {trend = \}, {trend = \\}, or "trend = \\". Do not use {trend = \}. To specify the inequality trend, use {trend!= }. For the std_error and slope expressions, you can also use the less-than sign (<).

The variation-aware analysis reports the results that meet any of the specified criteria.

This option is used with the **-analyze** option.

You can also set criteria for table bounds and slope analysis in this field.

```
{table_type lower_bound=lower_value upper_bound=upper_value |
table_type[table_sub_index] min_slope=min_value max_slope=max_value}
```

The table type, table sub_index, and associated group names are:

Type Name	Index Type	Related Group Names
delay	slew_index load_index	cell_rise cell_fall
transition	slew_index load_index	rise_transition fall_transition
constraint	constrained_index related_index	rise_constraint fall_constraint
energy	slew_index load_index	rise_power fall_power
current	time	output_current_rise output_current_fall
capacitance	slew_index load_index	receiver_capacitance1_rise receiver_capacitance1_fall receiver_capacitance2_rise receiver_capacitance2_fall
dc_current	input_voltage output_voltage	dc_current
ccsn	time	output_voltage_rise output_voltage_fall propagated_noise_low propagated_noise_high
pg_current	time	pg_current

{lvth_groups=lvth_name_list} Specifies a list of low threshold_voltage_group strings for power optimization, such as {lvth_groups="lvth_svt"}. lvth_groups is equivalent to the **-lvth_groups** option of the **set_multi_vth_constraint** command.

{clamping_ratio=ratio_value} Specifies a percentage for sensitivity analysis. By default clamping_ratio=0.95. This is used with -analyze {sensitivity} option.

-group_attribute {groups_or_attributes}

Checks or compares specific groups and attributes only.

You can specify one or more groups or attributes. An item can be a group, an attribute, a group and an attribute together with a value, or a group and an attribute together without a value. Use a space to separate each item in the list.

Use a forward slash (/) to combine a group name and an attribute name that you are specifying together. For example, the **{pin/direction}** option combines the pin group and the direction attribute under the pin group. In this case, the command checks the direction only in the pin group.

Use curly braces ({}) or double quotation marks ("") to enclose an expression delimited by spaces. The following example checks only the pin group whose name is A:

```
-group_attribute {"pin = A"}
```

The following wildcard characters are available for pattern matching:

- An asterisk (*) matches any string, including a null string.
- A question mark (?) matches any single character.

For example:

- **-group_attribute {pin}** checks the pin group only.
- **-group_attribute {direction}** checks the direction attribute only.
- **-group_attribute {pin/direction}** checks the pin's direction only.
- **-group_attribute {pin/direction=input}** checks the pin direction only when it is an input pin.
- **-group_attribute {cell_rise cell_fall output_current_*}** checks and validates the NLDM and CCS timing models only.
- **-group_attribute {va_* compact_ccs_*}** checks and validates the variation-aware timing models only.

If you do not specify the **-group_attribute** option, the **check_library** command checks all groups and attributes that apply to the specified mode.

```
-report_format {format_spec}
```

Specifies the format for the generated report. The syntax of the specification is as follows:

```
[csv[=csv_dir]] [nosplit] [html] [display] [sort_by_cell | sort_by_group_type]
```

You must include at least one of the following values in the specification:

- **csv[=csv_dir]**
Selects comma-separated values (CSV) as the report format, which is compatible with spreadsheet programs. By default, the CSV files are stored in the current working directory. Specify the directory to store the CSV files by using the *csv_dir* argument.

If you perform multiple runs with the same settings, the directory for the previous run is saved to a directory named *csv_dir_bak*.

If you do not specify this value, the **check_library** command generates a plain ASCII report.
- **nosplit**
Prevents line-splitting, and facilitates writing scripts to extract information from the report output. Most of the information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

- **html**[=*html_dir*]
Selects HTML as the report format. Optionally specify the directory in which to store the HTML files by using the *html_dir* argument. By default, it is the current directory.
- **display**
Launches a browser to view the check results in HTML. This should be used only with the **html** report format.
- **sort_by_cell** | **sort_by_group_type**
Specifies the sorting method. Use **sort_by_cell** to sort the validation and analysis tables by cell names. Use **sort_by_group_type** (the default) to sort the validation and analysis tables by group types, such as delay, slew, constraints, and receiver capacitance.

If you do not specify this option, the **check_library** command generates an ASCII report that is sorted by the group type.

-leq

Checks the logical equivalence for "when" conditions. For example, if you set the **-leq** option, the following expressions are reported as identical:

```
when : A+B
when : B+A
```

By default, logical equivalence is disabled and string comparisons are used.

-by_group_order

Compares or checks libraries by the order of library groups for non-unique groups and groups without a name, such as *leakage_power*. This is for concurrent checking of multiple libraries.

-char_integrity

Checks the following options for CCSN accuracy:

```
-validate { timing noise }
-analyze { sensitivity voltage_range }
```

-significant_digits *digits*

Specifies the number of significant digits displayed for floating-point numbers. Allowed values are from 0 through 13. The default value is 6.

-physical

Includes the following physical checking options:

```
-tech_consistency
-view_comparison
-same_name_cell
-signal_em
-antenna
-rectilinear_cell
-physical_only_cell
-phys_property {place route}
-rail view_data
-routeability
-tech
-drc
```

-logic_vs_physical

Checks all logic versus physical library checking, including the default checks for missing cells, missing or mismatched pins in the logic or physical library, and the following checks:

- cell_area
- cell_footprint
- bus_delimiter

-logic

Checks the following options for the logic library:

- scaling
- mcmm
- upf
- optimization {power}

-reset

Resets the options of the command to their default. Missing cells and missing or mismatched pins are checked between logic and physical libraries. All other options are ignored if used with the **-reset** option.

-all

Checks the following options for the library:

- tech_consistency
- view_comparison
- same_name_cell
- signal_em
- antenna
- rectilinear_cell
- physical_only_cell
- phys_property {place route cell metal_density}
- rail view_data
- routeability
- tech
- drc
- logic_vs_physical
- logic

DESCRIPTION

The **set_check_library_options** command sets specific options for the **check_library** command for logic library versus logic library checking, logic library versus physical library checking, and for checks between physical libraries.

Run the **set_check_library_options** command before you run the **check_library** command. If you do not specify any option with the **set_check_library_options** command, by default the command checks missing cells, missing and mismatched pins, including pg_pins versus power and ground pins.

The command returns the status, indicating success or failure of the option settings.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, the **-cell_footprint** and **-bus_delimiter** options are set, the lib1.db and lib2.db logic libraries are checked against the phys_lib physical libraries for missing cells and pins and mismatched pins. The cell footprints are compared, and the bus delimiters are checked:

```
prompt> set_check_library_options -cell_footprint \
      -bus_delimiter
1
prompt> check_library -mw_library_name {phys_lib} \
      -logic_library_name {lib1.db lib2.db}
1
```

```
#BEGIN_XCHECK_LIBRARY
```

```
Logic Library: lib1.db
              lib2.db
```

```
Physical Library: phys_lib
```

```
check_library options: -cell_footprint -bus_delimiter
```

```
Version: A-2007.12
```

```
Check date and time: Thu Jul 26 16:59:54 2007
```

```
#BEGIN_XCHECK_LOGICCELLS
```

```
Number of cells missing in logic library: 3 (out of 3348)
```

```
List of cells missing in logic library
```

Cell name	Cell type	Physical library
AND2	Core	phys_lib
NOR1	Core	phys_lib
XOR3	Core	phys_lib

```
List of physical only cells
```

Cell name	Cell type	Physical library
GFILL	Filler	phys_lib
GFILL10	Filler	phys_lib
FILL8	Filler	phys_lib

```
#END_XCHECK_LOGICCELLS
```

```
#BEGIN_XCHECK_PHYSICALCELLS
```

```
Number of cells missing in physical library: 0 (out of 846)
```

```
#END_XCHECK_PHYSICALCELLS
```

```
#BEGIN_XCHECK_PINS
```


Number of cells with missing or mismatched pins in libraries: 0

#END_XCHECK_PINS

#BEGIN_XCHECK_BUS

List of bus naming styles

Library name	Library type	Bus naming style

phys_lib	Physical library	_%d>
lib1.db	Logic library	

#END_XCHECK_BUS

#BEGIN_XCHECK_FOOTPRINT

Number of footprints: 1

List of cells with cell_footprint attribute

Footprint	Logic library name	Cell name	PR boundary

TIEH	lib1.db	GTIEH	(0,0)(0.8,1.8)
	lib1.db	TIEH	(0,0)(0.6,1.8)

#END_XCHECK_FOOTPRINT

Cross check summary:

Number of cells missing in logic library: 3

Logic library is INCONSISTENT with physical library.

#END_XCHECK_LIBRARY

1

SEE ALSO

- check_library(2)
- report_check_library_options(2)

set_cle_options

Sets command line editor settings.

SYNTAX

int **set_cle_options** [-mode *vi* / *emacs*] [-beep *on* / *off*] [-defaults]

ARGUMENTS

-mode "*vi* | *emacs*"

Sets the command line editor mode to either *vi* or *emacs*. Valid values are **vi** or **emacs**.

-beep "*on* | *off*"

Sets the command line editor beep sound to on or off. Valid values are **on** or **off**.

-defaults

Sets command line editor default settings. This option will override other options. Default edit mode is **emacs** and terminal beep is **off**.

DESCRIPTION

The **set_cle_options** command can be used to set command line editor settings.

If used without any options, it will display the current editor settings.

EXAMPLES

The following example sets command line editor to *vi* mode.

```
dc_shell-t> set_cle_options -mode vi
```

```
Information: Command line editor mode is set to vi successfully. (CLE-01)
```

```
1
```

The following example enables terminal beep.

```
dc_shell-t> set_cle_options -beep on  
1
```

The following example sets command line editor default settings.

```
dc_shell-t> set_cle_options -defaults  
1
```

The following example displays current command line editor settings.

```
dc_shell-t> set_cle_options  
  
*****  
Report : Command line editor settings  
Version: W-2004.12  
Date   : Mon Jan 31 13:02:38 2005  
*****
```

```
Information: dc_shell is currently in vi editing mode. (CLE-06)  
Information: Terminal beep is enabled. (CLE-07)
```

```
1
```

SEE ALSO

```
sh_enable_line_editing(3)  
sh_list_key_bindings(2)
```

set_clock_gate_latency

Specifies clock network latency values to be used for clock-gating cells, as a function of clock domain, clock-gating stage and fanout.

SYNTAX

```
status set_clock_gate_latency
  [-clock clock_list]
  [-overwrite]
  -stage cg_stage
  -fanout_latency cg_fanout_list
  [-transitive_fanout]
```

Data Types

```
clock_list    list
cg_stage      integer
cg_fanout_list string
```

ARGUMENTS

-clock *clock_list*

Specifies that the latency must be applied with respect to the specified clocks. If the **-clock** option is not specified, the latency is applied with respect to all clock domains to which the clock-gating cell belongs.

-overwrite

Specifies that clock latency values previously set on clock-gating cells should be overwritten.

-stage *cg_stage*

Specifies the clock-gating stage to which the clock latency data from the fanout range is applied. Registers are considered as stage 0.

-fanout_latency *cg_fanout_list*

Specifies the list of clock-gating cells fanout and clock latency values; for example, {1-5 0.9, 6-20 0.5, 21-inf 0.3}. A fanout of 1 to 5 has a latency of 0.9; a fanout of 21 or larger has a latency of 0.3. If the same latency value is wanted for the entire fanout range, it can be specified as {1-inf 0.9}.

-transitive_fanout

Specifies that transitive fanout should be used instead of direct fanout when annotating clock gate latency. Direct fanout is the default.

DESCRIPTION

The **set_clock_gate_latency** command allows you to specify clock network latency values for clock-gating cells, as a function of clock domain, clock-gating stage, and fanout. These latency values are annotated on the clock pins of clock-gating cells when running the **compile_ultra** command, or when running the **apply_clock_gate_latency** command.

The clock-gate latency settings specified using the **set_clock_gate_latency** command are stored as an attribute on the source port of each specified clock. Invoke this command after defining all clocks of the design. These stored latency values are annotated on the clock pins of clock-gating cells by means of one of the following processes:

- Using the **apply_clock_gate_latency** command, which applies latency on any existing clock-gating cells
- During **compile_ultra** command, which applies latency on any existing clock-gating cells or those inserted or modified during the compile

NOTE: The stages of the clock gating network are numbered from 0 for the register stage number and increasing toward the clock source. For example, in a four stage clock gating tree, Stage 0 drives register clocks directly, and Stage 3 is the clock gating cell with its input directly connected to the clock source.

Use the **set_clock_gate_latency** command to specify the latency for gated registers by specifying the command for stage 0. By definition stage 0 has no fanout, so use the following syntax:

```
prompt> set_clock_gate_latency -stage 0 -fanout_latency \
{ 1-inf value }
```

When clock latency settings are provided for stage 0, the values are annotated on the output pin (enabled clock pin) of each clock-gating cell that is directly driving gated registers. If no clock latency is set for stage 0, but a clock latency has been set on the clock object by using **set_clock_latency** command, then this value is used for clock latency. For ungated registers, the latency is propagated by the timer from the clock object latency.

You can use the **-clock** and **-overwrite** options with the **-stage 0** option.

Use the **-transitive_fanout** option to calculate latency value based upon the transitive fanout of a clock-gating cell instead of the direct fanout. The transitive fanout of a clock-gating cell is defined as the total number of all driven leaf registers. The registers driven by a stage 1 clock-gating cell, ICG_1, are counted as part of the transitive fanout of the stage 2 clock-gating cell, ICG_2, that drives ICG_1. By definition:

stage 2 transitive fanout >= stage 1 transitive fanout

NOTE: Transitive fanout only counts registers, unlike the direct fanout which counts both registers and driven clock-gating cells.

When the **-transitive_fanout** option is used for the first time, a *fanout mode* is set, and this is indicated by the PWR-916 information message. You cannot mix fanout modes of the **set_clock_gate_latency** command, where some instances use the **-transitive_fanout** option and others do not. If you want to switch between fanout modes, use the **reset_clock_gate_latency** command.

If you specify fanout ranges in the *cg_fanout_list* and you do not cover all values from 1 to inf, the fanout ranges with missing latency information are filled by the tool with the smallest latency value from the adjacent ranges.

In a clock tree, latency values are lowest for cells closest to the top of the tree (clock source). Transitive fanout values are highest for cells at the source and decrease toward the registers. Therefore, specified clock latency values must decrease with respect to fanout ranges otherwise a PWR-742 warning message is generated.

If clock latency settings are not specified for a certain clock-gating stage and a clock-gating cell within this stage exists, the latency is propagated from the driving clock-gating cell or from clock object latency, and a warning message is issued. These actions are performed during the annotation process:

apply_clock_gate_latency or **compile_ultra** command.

If specified clock latency values do not decrease with respect to clock gating stages, the PWR-744 warning message is generated. This check is disabled if the **-transitive_fanout** option is used. In addition, if an inconsistent clock latency annotation is detected when running the **apply_clock_gate_latency** or **compile_ultra** command, the tool attempts to fix it by assuming the smallest value from the clock latencies annotated on gated clock-gating cells or registers. This ensures that clock latencies are monotonically increasing values in the path going from the clock source to the gated registers. The following actions can be performed by the tool to achieve this:

- If clock-gating cell A is directly driving one or more registers, and some clock latency settings have been provided for the timing of these gated registers (by using value 0 for the **-stage** option of **set_clock_gate_latency** command or by propagation of latency specified for clock object), and the latency annotated on cell A is higher than the latency provided for gated registers, then the latency on clock-gating cell A is overwritten with the latency provided for gated registers. If this fix is done, the PWR-746 warning message is generated.
- If clock-gating cell A is driving another clock-gating cell B, and the clock latency set on A is higher than the one set on B, the tool resolves it by overwriting the clock latency on A with the value set on B. If this fix is done, the PWR-746 warning message is generated.

To remove the settings specified using **set_clock_gate_latency**, use the **reset_clock_gate_latency** command.

If the variable **power_cg_physically_aware_cg** is enabled, then the annotation of latency values from **set_clock_gate_latency** is disabled.

Multicorner-Multimode Support

This command is scenario dependent and affects the current scenario only.

The actual clock latency annotation performed during the execution of the **compile_ultra** or **apply_clock_gate_latency** commands is done for all the scenarios for which a clock latency value is available.

EXAMPLES

The following example specifies the clock latency values for the complete fanout range of clock-gating cells for stages 1, 2, and 3. This latency data applies to the clock-gating cells whose clock pins belongs to clock *clk1*.

```
prompt> set_clock_gate_latency -clock [get_clocks clk1] -stage 1 \
      -fanout_latency {1-30 2.1, 31-100 1.7, 101-inf 1.1}
```

```
prompt> set_clock_gate_latency -clock [get_clocks clk1] -stage 2 \
      -fanout_latency {1-5 0.9, 6-20 0.5, 21-inf 0.3}
```

```
prompt> set_clock_gate_latency -clock [get_clocks clk1] -stage 3 \
      -fanout_latency {1-10 0.28, 11-inf 0.11}
```

In the following example, fanout ranges specified in the **-fanout_latency cg_fanout_list** do not cover all values from 1 to inf. In this case, the tool assumes a clock latency value of 1.7 to be applied to fanout range 11-30 and a clock latency of 1.1 for fanouts higher than 80.

```
prompt> set_clock_gate_latency -stage 1 \
      -fanout_latency {1-10 2.1, 31-80 1.7, 101-300 1.1}
```

The following example shows how to specify and annotate different clock latency values using the transitive fanout.

```
prompt> set_clock_gate_latency -clock [get_clocks clk1] -stage 1 \
```

```
-fanout_latency {1-30 2.1, 31-100 1.7, 101-inf 1.1} \  
-transitive_fanout
```

```
prompt> set_clock_gate_latency -clock [get_clocks clk1] -stage 2 \  
-fanout_latency {1-5 0.9, 6-20 0.5, 21-inf 0.3} \  
-transitive_fanout
```

The following example shows how to specify and annotate different clock latency values for different multicorners-multimode scenarios.

```
prompt> current_scenario sc1  
prompt> set_clock_gate_latency -stage 1 \  
-fanout_latency {1-inf 0.11}
```

```
prompt> current_scenario sc2  
prompt> set_clock_gate_latency -stage 1 \  
-fanout_latency {1-inf 0.15}
```

```
prompt> compile_ultra -gate_clock
```

SEE ALSO

```
apply_clock_gate_latency(2)  
power_cg_physically_aware_cg(3)  
remove_clock_latency(2)  
report_clock_gating(2)  
reset_clock_gate_latency(2)  
set_clock_latency(2)
```

set_clock_gating_check

Puts setup and hold checks on clock-gating cells.

SYNTAX

```
status set_clock_gating_check
  [-setup setup_margin]
  [-hold hold_margin]
  [-rise]
  [-fall]
  [-high | -low]
  [object_list]
```

Data Types

```
setup_margin  float
hold_margin   float
object_list   collection
```

ARGUMENTS

-setup *setup_margin*

Specifies the setup margin for the clock-gating signal. For AND and NAND clock-gating elements, the setup time is checked relative to the 0->1 transition of the clock input. For OR and NOR clock-gating elements, the setup time is checked against the 1->0 transition of the clock input. For more complicated clock-gating elements, the setup check is against the clock transition where the clock input goes from a controlling value to a noncontrolling value for the clock-gating element.

-hold *hold_margin*

Specifies the hold margin for the clock-gating signal. For AND and NAND clock-gating elements, the hold time is checked relative to the 1->0 transition of the clock input. For OR and NOR clock-gating elements, the hold time is checked against the 0->1 transition of the clock input. For more complicated clock-gating elements, the hold check is against the clock transition where the clock input goes from a noncontrolling value to a controlling value for the clock-gating element.

-rise

Specifies if rising delays are constrained for clock-gating checks. If you do not specify the **-rise** or **-fall** option, both rising and falling delays are constrained.

-fall

Specifies if falling delays are constrained for clock-gating checks. If you do not specify the **-rise** or **-fall** option, both rising and falling delays are constrained.

-high

Indicates that the check is to be performed on the high level of the clock. By default, timing analysis determines whether to use the high or low level of the clock using information from the cell's logic. That is, for AND and NAND gates, the check is performed on the high level; for OR and NOR gates, on the low level. For some complex cells (for example, MUX, OR-AND) it cannot be determined which level to use, and so no check is performed unless either **-high** or **-low** is specified. This option can be used only when the object list contains cells or pins.

-low

Indicates that the check is to be performed on the low level of the clock. By default, timing analysis determines whether to use the high or low level of the clock using information from the cell's logic. That is, for AND and NAND gates, the check is performed on the high level; for OR and NOR gates, on the low level. For some complex cells (for example, MUX, OR-AND) it cannot be determined which level to use, and so no check is performed unless either **-high** or **-low** is specified. This option can be used only when the object list contains cells or pins.

object_list

Specifies the cells, pins, or clock objects where the clock-gating setup or hold margins are to be checked.

If you do not specify the *object_list* argument, the clock-gating check is applied to the current design.

To enable clock-gating checks only for specific clock-gating cells, specify the names of the cells in the object list. Clock-gating checks for specific cells override any clock-gating checks that have been specified for the design.

To enable clock-gating checks only for specific pins, specify the pins in the object list.

To enable clock-gating checks on all clock-gating cells driven by specific clocks, specify the clock objects in the object list.

DESCRIPTION

The **set_clock_gating_check** command provides the ability to check setup or hold margins for control inputs of clock-gating cells. The setup check ensures that the clock-gating input stabilizes for a given amount of time before the clock input of the gating cell makes a transition to a noncontrolling value. The hold check ensures that the clock-gating signal stabilizes for a given amount of time after the clock input has gone back to a controlling value. Together, the two checks ensure that the clock-gating signal stabilizes for the entire period of time when the gated clock input has a noncontrolling value. This is so that the gating signal cannot "clip" a clock edge or generate spurious clock pulses by changing when the clock input is noncontrolling.

You can specify in which interval of the clock the clock-gating checks are made by using the **-high** and **-low** options.

The clock-gating checks created by **set_clock_gating_check** act as optimization constraints; the tool tries to adjust the delays of the logic driving the clock-gating inputs to avoid setup or hold violations.

Be aware that the presence of clock-gating checks on a clock-gating cell disables certain logic transformations that otherwise might be applied to the cell. The tool can adjust only the size of the cell by replacing it with other cells with the same logic function but with different drive capacity. No other logic transformations involving the clock-gating cell are permitted.

The **set_clock_gating_check** command can create setup and hold checks only against a clock signal. The command cannot introduce setup or hold checks between two nonclock signals.

To undo **set_clock_gating_check**, use the **remove_clock_gating_check** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example creates setup checks with a 0.75 margin and hold checks with a 0.5 margin for all clock-gating inputs in the current design:

```
prompt> set_clock_gating_check -setup 0.75 -hold 0.5
```

The following example creates setup checks with a 0.58 rise margin for all clock-gating cells driven by the CLK1 clock:

```
prompt> set_clock_gating_check -setup 0.58 -rise [get_clocks CLK1]
```

The following example creates hold checks with a margin of 0.64 on all gating inputs of the CLK_GATE1 cell:

```
prompt> set_clock_gating_check -hold 0.64 CLK_GATE1
```

The following example creates a setup check with a margin of 0.44 and a hold check with a margin of 0.64 on all gating inputs of the CLK_GATE1 cell and specifies that the clock is noncontrolling when it is low:

```
prompt> set_clock_gating_check -setup 0.44 -hold 0.64 -low CLK_GATE1
```

SEE ALSO

- `create_clock(2)`
- `current_design(2)`
- `remove_clock_gating_check(2)`
- `report_clock(2)`

set_clock_gating_enable

Controls the signals to be used as the clock gating enable in the execution of the **compile_ultra -gate_clock** command.

SYNTAX

```
status set_clock_gating_enable  
[-exclude objects_to_exclude]  
[-undo objects_to_remove_exclusion]
```

Data Types

```
objects_to_exclude      collection  
objects_to_remove_exclusion collection
```

ARGUMENTS

-exclude *objects_to_exclude*

Specifies a collection of objects in the current design that are considered the source of a signal that must be excluded (if logically possible) from the enable logic for clock gating. Objects can be of following types:

- Primary input ports
- Sequential cells, black boxes, or macros
- Output pins of sequential cells, black boxes, or macros

-undo *objects_to_remove_exclusion*

Removes from the given objects the directives specified by a previously executed **set_clock_gating_enable -exclude** command. Objects can be of following types:

- Primary input ports
 - Sequential cells, black boxes, or macros
 - Output pins of sequential cells, black boxes, or macros
-

DESCRIPTION

This command provides a mechanism to tell the tool that a given signal should not be used as an enable condition for clock gating.

During the execution of the **compile_ultra -gate_clock** command, the tool looks at the transitive fanin of the logic cone driving the

enable signal for clock gating (for both preexisting clock gates and candidate registers). If any object is found with the exclusion constraint, the tool tries to compute a modified enable expression without dependency of this excluded signal. The clock-gating enable signal and register's enable pin signal are modified for this purpose. If it is feasible to compute those signals, the enable logic is modified accordingly. If signal exclusion is logically infeasible, the candidate registers are not clock gated and the preexisting clock gates are removed.

A register is also not clock gated if the synthesis of the modified enable signal requires modifying the boundary of a constrained hierarchy.

The registers that are not clock gated because of this inability of excluding a certain signal are reported with the message **Suitable enable signal has been excluded** by the **report_clock_gating -ungated** command.

The Tcl variable **power_cg_all_registers** or the use of **set_clock_gating_objects -force_include** command does not prevent the removal of clock gates with infeasible signal exclusions. In both cases, the registers are gated by an always-enabled clock gate.

The presence of enable exclusion directives on any object has no impact in the computation of enable signals for XOR Self Gating.

You must run the **set_clock_gating_enable** command before running the **compile_ultra -gate_clock** command. The specifications persist during the subsequent executions of **compile_ultra -gate_clock**. The directives are also stored in the .ddc file and written by the **write_script** command (unless **-no_cg** option is used) for ASCII flow support.

One and only one of the two arguments must be used for the command to succeed.

This command can be used as part of a Design Compiler Embedded Tcl script on the RTL.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example excludes input port **en** as a valid enable signal for clock gating:

```
prompt> set_clock_gating_enable -exclude [get_pots en]
```

In the following example, all the output pins of register **reg** are discarded as potential drivers of an enable signal for clock gating:

```
prompt> set_clock_gating_enable -exclude [get_cells reg]
```

The following example restores input port **en** as a valid driver of enable signal for clock gating:

```
prompt> set_clock_gating_enable -undo [get_pots en]
```

SEE ALSO

- `compile_ultra(2)`
- `remove_clock_gating(2)`
- `report_clock_gating(2)`
- `set_clock_gating_objects(2)`
- `set_clock_gating_style(2)`
- `write_script(2)`

set_clock_gating_objects

Forces the enabling or disabling of clock gating for specified objects in the current design, overriding all conditions necessary for automatic RTL clock gating, by the **compile_ultra -gate_clock** command. Objects can be of type register, hierarchical cell, power domain, or design.

SYNTAX

```
status set_clock_gating_objects  
  [-force_include object_list]  
  [-exclude object_list]  
  [-include object_list]  
  [-undo object_list]
```

Data Types

object_list list

ARGUMENTS

-force_include *object_list*

Specifies a list of objects in the current design that to be clock gated, overriding the conditions set by the clock-gating style. The *object_lists* is mutually exclusive, that is, an object cannot be specified with more than one of the following options: -**force_include**, -**exclude**, -**include** and -**undo**.

-exclude *object_list*

Specifies a list of objects in the current design to be excluded from clock gating. The *object_lists* are mutually exclusive, that is, an object cannot be specified with more than one of the following options: -**force_include**, -**exclude**, -**include** and -**undo**.

-include *object_list*

Specifies a list of objects in the current design for which clock gating should be done as per the style specified by the **set_clock_gating_style** command. This is the default for all objects in the design. An object cannot be specified with more than one of the following options: -**force_include**, -**exclude**, -**include** or -**undo**.

-undo *object_list*

Specifies a list of objects in the current design for which the inclusion or exclusion criteria for clock gating should be removed. The *object_lists* are mutually exclusive. An object cannot be included on more than one of the following options: -**force_include**, -**exclude**, -**include** and -**undo**.

DESCRIPTION

This command specifies the objects on which clock gating is to be either enabled or disabled on its registers, overriding all conditions necessary for automatic clock gating by the **compile_ultra -gate_clock** command.

For objects specified with the **-force_include** option, clock gating is performed regardless of the previous specification. Objects specified with the **-exclude** option are excluded from clock gating regardless of the previous specification.

Use the **-include** option to specify that regular clock gating criteria should be used on the specified objects.

Use the **-undo** option to remove a previously specified criteria on the specified objects.

You must run the **set_clock_gating_objects** command before running the **compile_ultra -gate_clock** command. The specifications persists during the subsequent executions of the **compile_ultra -gate_clock**.

This command can be used as part of a DC Embedded Tcl script on the RTL.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example excludes the register bank A_reg in the current design, from clock gating:

```
prompt> set_clock_gating_objects -exclude A_reg[*]
```

In the following example, the register bank D_OUT_reg in the MID subdesign, is clock-gated, ignoring the previous specification.

```
prompt> set_clock_gating_objects -force_include MID/D_OUT_reg[*]
```

The following example includes and excludes certain registers from clock gating in the ADDER subdesign:

```
prompt> set_clock_gating_objects \  
-force_include ADDER/out1_reg[*] \  
-exclude ADDER/out2_reg[*]
```

The following example removes the directive to include and exclude clock gating for the registers in the ADDER subdesign:

```
prompt> set_clock_gating_objects \  
-undo {ADDER/out1_reg[*] ADDER/out2_reg[*]}
```

The following example excludes every register present in the ADDER subdesign, except for the out1_reg bank, which is clock gated according to the specified style.

```
prompt> set_clock_gating_objects \  
-exclude ADDER \  
-include ADDER/out1_reg[*]
```

SEE ALSO

compile(2)
compile_ultra(2)
set_clock_gating_style(2)
report_clock_gating(2)
remove_clock_gating(2)

set_clock_gating_registers

Forces the enabling or disabling of clock gating for the specified registers in the current design, overriding all conditions necessary for automatic RTL clock gating, performed by the **compile_ultra -gate_clock** command.

SYNTAX

```
status set_clock_gating_registers  
  [-include_instances register_list]  
  [-exclude_instances register_list]  
  [-undo register_list]
```

Data Types

register_list list

ARGUMENTS

-include_instances *register_list*

Specifies a list of registers in the current design to be included for clock gating. A register cannot be specified with more than one of the **-include_instances**, **-exclude_instances**, and **-undo** options.

-exclude_instances *register_list*

Specifies a list of registers in the current design to be excluded from clock gating. A register cannot be specified with more than one of the **-include_instances**, **-exclude_instances**, and **-undo** options.

-undo *register_list*

Specifies a list of registers in the current design for which the previous specification of the **set_clock_gating_registers** command is to be undone. A register cannot be specified with more than one of the **-include_instances**, **-exclude_instances**, and **-undo** options.

DESCRIPTION

The **set_clock_gating_registers** command will be obsolete in a future release. Use the **set_clock_gating_objects** command instead, to specify inclusion criteria for different registers.

This command specifies registers for which clock gating is to be either enabled or disabled, overriding all conditions necessary for automatic clock gating by the **compile_ultra -gate_clock** command. Registers specified with the **-include_instances** and **-exclude_instances** options are clock-gated or excluded from clock-gating respectively, regardless of any previous specifications.

Use the **-undo** option to remove the specification already set by the **set_clock_gating_registers** command.

You must run the **set_clock_gating_registers** command before running the **compile_ultra -gate_clock** command. The specification is used during all subsequent executions of **compile_ultra -gate_clock** command.

The **set_clock_gating_registers** command directs the **compile_ultra -gate_clock** command to implement synchronous load enable flip-flops with gated clocks instead of data feedback through multiplexers, if certain conditions (for example, minimum register bank size) are satisfied. This command overrides these conditions for the specified registers; thus, registers specified by the **-include_instances** option are clock gated, and registers specified by the **-exclude_instances** option are not clock gated, regardless of any previously-specified conditions necessary for automatic clock gating.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example excludes the register bank named *A_reg* in the current design, so that these registers are not clock gated:

```
prompt> set_clock_gating_registers -exclude_instances A_reg[*]
```

The following example includes the register bank named *D_OUT_reg* in the *MID* subdesign, so that this register is always clock gated:

```
prompt> set_clock_gating_registers -include_instances MID/D_OUT_reg[*]
```

The following example includes and excludes clock gating for registers in the *ADDER* subdesign:

```
prompt> set_clock_gating_registers \  
-include_instances ADDER/out1_reg[*] \  
-exclude_instances ADDER/out2_reg[*]
```

The following example removes the directive to include and exclude clock gating to and from the registers in the *ADDER* subdesign:

```
prompt> set_clock_gating_registers \  
-undo {ADDER/out1_reg[*] ADDER/out2_reg[*]}
```

SEE ALSO

compile(2)
compile_ultra(2)
set_clock_gating_style(2)
set_clock_gating_objects(2)
report_clock_gating(2)

set_clock_gating_style

Sets the clock-gating style for clock-gate insertion and replacement.

SYNTAX

```
status set_clock_gating_style
  [-sequential_cell none | latch]
  [-minimum_bitwidth minsize_value]
  [-setup setup_value]
  [-hold hold_value]
  [-positive_edge_logic {cell_list | integrated [active_low_enable] [invert_gclk]]}
  [-negative_edge_logic {cell_list | integrated [active_low_enable] [invert_gclk]]}
  [-control_point none | before | after]
  [-control_signal scan_enable | test_mode]
  [-observation_point true | false]
  [-observation_logic_depth depth_value]
  [-max_fanout max_fanout_count]
  [-num_stages num_stages_count]
  [-no_sharing]
  [-instances {instances_list}]
  [-power_domains {power_domain_list}]
  [-designs {designs_list}]
```

Data Types

<i>minsize_value</i>	integer
<i>setup_value</i>	float
<i>hold_value</i>	float
<i>depth_value</i>	integer
<i>max_fanout_count</i>	integer
<i>num_stages_count</i>	integer
<i>instances_list</i>	list
<i>power_domain_list</i>	list
<i>designs_list</i>	list

ARGUMENTS

-sequential_cell none | latch

Specifies the cell that is used to delay the enable signal gating the clock. By default, the tool selects a latch cell. To select a latch-free style, use **none**. You can specify a specific latch cell by using the following syntax:

```
latch:lib_cell
```

The latch style must be compatible with the clock-gating circuitry as chosen by the **-positive_edge_logic** and **-negative_edge_logic** options. If you need to use different latches for positive-edge logic and negative-edge logic, use the **-positive_edge_logic** and **-negative_edge_logic** options, instead of the **-sequential_cell** option. If you specify either of the following, the tool infers a latch-based clock-gating style:

```
-sequential_cell none -positive_edge_logic {latch and}
-sequential_cell none -negative_edge_logic {latch or}
```

-minimum_bitwidth *minsize_value*

Specifies the minimum size of a register bank for which the clock can be gated.

-setup *setup_value*

Specifies the setup time constraint at the enable input of the 2-input clock gate in the clock-gating circuitry.

-hold *hold_value*

Specifies the hold time constraint at the enable input of the 2-input clock gate in the clock-gating circuitry.

-positive_edge_logic { *cell_list* | **integrated active_low_enable invert_gclk** }

Specifies the circuitry used to gate the clock of a flip-flop that is inferred by a positive edge clock construct in the HDL code. You can specify the circuitry in one of the following ways:

- *cell_list*
Specifies a list of 1 to 4 strings that describe the gates inserted in the clock network. The circuitry must be compatible with the latch style specified by the **-sequential_cell** option. For example, AND or NAND functionality for a latch-based style and OR or NOR functionality for a latch-free style.

If the circuitry inverts the clock signal, an additional inverter is inferred between the clock output pin of the clock-gating circuitry and the clock input pin of the register. You can map this inverter with the generic sequential cell to a negative-edge triggered flip-flop.
- {integrated [active_low_enable] [invert_gclk]}
Uses a single special integrated cell instead of the clock-gating circuitry.

-negative_edge_logic { *cell_list* | **integrated active_low_enable invert_gclk** }

Specifies the circuitry used, to gate the clock of a flip-flop that is inferred by a negative-edge clock construct in the HDL code. You can specify the circuitry in one of the following ways:

- *cell_list*
Specifies a list of 1 to 4 strings that describe the gates inserted in the clock network. The circuitry must be compatible with the latch style specified by the **-sequential_cell** option. For example, OR, NOR functionality for a latch-based style and AND, NAND functionality for a latch-free style.

If the circuitry does not invert the clock signal, an additional inverter is inferred between the clock output pin of the clock-gating circuitry and the clock input pin of the register. In contrast to an inverter specified in the clock-gating circuitry, you can map this inverter with the generic sequential cell to a negative-edge triggered library flip-flop.
- {integrated [active_low_enable] [invert_gclk]}
Uses a single special integrated cell instead of the clock-gating circuitry.

-control_point none | before | after

Specifies where to insert a control point in the clock gating circuitry. With **before** or **after** control styles, the tool implements an OR gate with the original register enable signal and a test control signal as inputs. If you specify a latch-based style, the **before** or **after** control styles determine the location of the OR gate with respect to the latch. If you specify a latch-free style, the **before** and **after** control styles are equivalent. The tool creates a new input port to provide the test signal. The control points must be hooked up to the design level test_mode or scan_enable port using the **insert_dft** command.

-control_signal scan_enable | test_mode

Specifies the test control signal. If an input port is created and the argument of the **-control_signal** option is *scan_enable*, the name of the port is determined by the **test_scan_enable_port_naming_style** variable, and a **test_scan_enable** signal_type attribute is put on the new port. If an input port is created and the argument of the **-control_signal** option is *test_mode*, the name of the port is determined by the **test_mode_port_naming_style** variable, and a **test_hold** attribute is set on the new port.

-observation_point true | false

Specifies whether or not to implement observability logic. The observed signals are the enable signal inputs to the control point OR gates. A new input port is created to provide the test signal, which blocks activity to the observability logic during mission mode.

-observation_logic_depth depth_value

Specifies the maximum depth of an XOR tree built in the observability circuitry.

-max_fanout max_fanout_count

Specifies the maximum size of a register bank which can be gated with a clock gate.

-num_stages num_stages_count

Specifies the maximum number of stages for multistage clock gating.

-no_sharing

Limits the registers gated by a clock-gating cell to a single register bank.

-instances {instances_list}

Specifies that the clock-gating style should be applied over the given list of hierarchical cells.

-power_domains {power_domain_list}

Specifies that the clock-gating style should be applied over the given list of power domains.

-designs {designs_list}

Specifies that the clock-gating style should be applied over the given list of designs.

DESCRIPTION

This command sets the clock-gating style to be used for clock gating with the **compile_ultra -gate_clock** and **replace_clock_gates** commands, as well as gate-level clock gating enabled by specifying the **-gate_clock** option to the **compile** or **compile_ultra** command.

Clock gating of register banks is an alternative implementation of load-enable registers. In comparison to the traditional implementation using recirculating feedback and multiplexers, power consumption is reduced on the clock tree, in the registers, and in the combinational logic (multiplexers). Clock gating has implications on testability and physical design.

The clock-gating style specifies three aspects of clock gating:

- The conditions when clock gating is applied
- The clock-gating circuitry that is implemented
- Additional circuitry to improve testability.

Clock-gating Conditions

A number of conditions must be satisfied to gate a register clock in a latch-based style. First, the register must be a load enable flip-flop. Second, the register belongs to a bank with a bit width equal to or larger than the value specified by the **-minimum_bitwidth** option. Because clock-gating circuitry, and possibly some testability-improving gates, is implemented for each register bank, a register bank should be a reasonable size to be considered for clock gating. The default is 3. An additional condition (known as the setup condition) must be satisfied for latch-free clock gating. All of the inputs to the combinational logic generating the load enable signal of a register bank must be flip-flops that are triggered by the same clock, and on the same clock edge as the register bank. For hierarchical designs, it might be necessary to derive the related clock for the inputs to the combinational logic from the clock connectivity in parent hierarchies. This is enabled by setting the **power_cg_derive_related_clock** variable to **true**. Use the **set_clock_gating_objects** command to force clock gating of registers that do not satisfy these conditions, or disable the setup condition for all registers by setting the **power_cg_ignore_setup_condition** variable to **true**.

Specifying the Clock-Gating Circuitry

The clock-gating circuitry is a hierarchical cell. The cell is attributed such that its structure is maintained during compilation, which means you have tight control over the structure of the clock-gating circuitry. There are several options to determine the clock-gating circuitry. The most important option is **-sequential_cell**, which determines latch-based or latch-free clock gating. The basic effect of latch-based clock gating is to prevent a leading clock edge by maintaining the value of the clock signal after the trailing edge. For example, for positive-edge triggered flip-flops, the clock signal is forced and kept to Logic0 after the negative edge, preventing a rising edge that loads new data into the register. To achieve this, the clock-gating circuitry must have AND or NAND functionality for positive-edge triggered flip-flops, and OR or NOR functionality for negative-edge triggered flip-flops. To prevent problems on the gated clock signal, the load enable signal is delayed by a latch that is transparent and it is between the trailing and the leading clock edge. Additionally, setup and hold time constraints are set at the enable input of the clock gate. The **-setup** option specifies the setup time value; the **-hold** option specifies the hold time value. The **-sequential_cell** option can also be used to specify a particular D-latch from a particular library:

```
string seq_cell = none | latch[:[library_name]/cell_name]
string library_name
string cell_name
```

Latch-free clock gating prevents a leading clock edge by maintaining the value of the clock signal before the trailing edge. For example, for positive-edge triggered flip-flops, the clock signal is forced and kept to Logic1 before the negative edge. To achieve this, the clock-gating circuitry must have OR or NOR functionality for positive-edge triggered flip-flops, and AND or NAND functionality for negative-edge triggered flip-flops. Because the load enable signal must arrive before the trailing edge of the clock, there is no latch. Additionally, setup and hold time constraints are set at the enable input of the clock gate. The setup time value is specified by the **-setup** option; the hold time value is specified by the **-hold** option.

Before compilation, you must propagate the setup and hold timing constraints to the current design using the **propagate_constraints-gate_clock** command.

Note that the term "positive-edge triggered flip-flop" here means that the register is inferred by a positive edge clock construct in the HDL code. Because the clock-gating circuitry for positive-edge triggered flip-flops differs from that for negative-edge triggered flip-flops, you have two options for specifying the clock gating circuitry: **-positive_edge_logic**, for flip-flops inferred by a positive edge construct in the HDL code and **-negative_edge_logic**, for flip-flops inferred by a negative edge construct in the HDL code. With each of these options, you can specify a list of one to four strings that describe the gates inserted in the clock network. If you specify four strings, the first string specifies the latch, the second string specifies a 1-input gate (inverter or buffer) in the fanin of the clock gate on the clock network, the third string specifies the 2-input clock gate (AND/NAND/OR/NOR gate), and the fourth string specifies a 1-input gate (inverter or buffer) in the fanout of the clock gate. The specification of the 1-input gates in the fanin and fanout of the clock gate is optional. There can be four strings, each of which can contain the specification of a library cell:

```
string gate = gate_type[:[library_name]/cell_name]
string gate_type = latch | and | nand | or | nor | buf | inv
string library_name
string cell_name
```

Note that the **-positive_edge_logic** and **-negative_edge_logic** options must be compatible with the latch style as specified by the

-sequential_cell option or in the gate-level list. For latch-based clock gating, the **-positive_edge_logic** option must specify AND or NAND functionality, and the **-negative_edge_logic** option must specify OR or NOR functionality. For latch-free clock gating, the **-positive_edge_logic** option must specify OR or NOR functionality, and the **-negative_edge_logic** option must specify AND or NAND functionality.

The AND functionality is given by one of the following lists:

```
{and}
{buf and}
{and buf}
{buf and buf}
{nand inv}
{buf nand inv}
{inv or inv}
{inv nor}
{inv nor buf}
```

The NAND functionality is given by one of the following lists:

```
{and inv}
{buf and inv}
{nand}
{buf nand}
{nand buf}
{buf nand buf}
{inv or}
{inv or buf}
{inv nor inv}
```

The OR functionality is given by one of the following lists:

```
{or}
{buf or}
{or buf}
{buf or buf}
{nor inv}
{buf nor inv}
{inv and inv}
{inv nand}
{inv nand buf}
```

The NOR functionality is given by one of the following lists:

```
{or inv}
{buf or inv}
{nor}
{buf nor}
{nor buf}
{buf nor buf}
{inv and}
{inv and buf}
{inv nand inv}
```

You can designate certain library cells to be used exclusively or preferentially for gating the clocks. Such cells can be used for the 2-input clock gate, inverters, buffers, or D-latches in the clock-gating circuitry. To use a specific cell for clock gating and to prevent its use in other areas of the design, set the **dont_use** and the **is_clock_gating_cell** attributes to **true** in the library description. To preferentially use a specific cell in the clock-gating circuitry, set only the **is_clock_gating_cell** attribute to **true**. The **is_clock_gating_cell** attribute is a user-defined attribute of type Boolean for the cell group.

You can also designate a specific input pin of the 2-input clock gate as the enable input. If the clock attribute is set on one of the input pins of the 2-input clock gate, the other input is recognized as the enable pin. Otherwise, if the **clock_gate_enable_pin** attribute is set to **true** on an input, this pin is recognized as the enable pin. The **clock_gate_enable_pin** attribute is a user-defined attribute of type Boolean for the pin group.

Besides specifying simple 2-input gates, you also have the option to specify that a single complex clock-gating cell be used for the clock-gating logic. This is done by specifying **integrated** for the gate list to the **-positive_edge_logic** or **-negative_edge_logic** options. Note that if these options are specified as **integrated**, the strings after it are of integrated clock-gating cell types. No inverters or buffers are allowed before or after the integrated cell. This string can contain the specification of a library cell:

```
string gate = {integrated[:{library_name}/cell_name] \
               [active_low_enable] [invert_gclk]}
string library_name
string cell_name
```

If a library cell is specified for use as an integrated clock-gating cell, the cell must have the **clock_gating_integrated_cell** attribute. Specify the attribute value as **generic**. The Library Compiler tool infers the functionality of the library cell based on the logic functions of the pins of the library cell. For backward compatibility, this string attribute can still have the value and specifies the functionality of the cell as a clock-gating cell. Based on the presence or absence of a latch, whether it is to be used as **positive_edge_logic** or **negative_edge_logic**, whether the control point is present and whether the observability port is present, the **clock_gating_integrated_cell** can have the following values:

```
latch_posedge
latch_posedge_precontrol
latch_posedge_postcontrol
latch_posedge_precontrol_obs
latch_posedge_postcontrol_obs
latch_negedge
latch_negedge_precontrol
latch_negedge_postcontrol
latch_negedge_precontrol_obs
latch_negedge_postcontrol_obs
none_posedge
none_posedge_control
none_negedge
none_negedge_control
```

Also, the pins of the integrated cell must have appropriate attributes, which include,

```
clock_gate_enable_pin attribute on the enable input pin
clock_gate_clock_pin attribute on the clock input pin
clock_gate_out_pin attribute on the gated-clock output pin
clock_gate_test_pin attribute on the scan-enable or test-mode pin
clock_gate_obs_pin attribute on the observability output pin
```

Specifying Additional Circuitry to Improve Testability

The next sections describe the specification of additional circuitry to improve testability of designs with gated clocks. The last four options specify circuitry that is introduced to improve testability. The impact of clock gating on testability is twofold. First, due to the clock gate, the control of the gated clock signal is reduced. Therefore, the clock-gating style allows the insertion of an OR gate, which forces the enable signal to logic 1 during testing. The **-control_point** option specifies if and where the OR gate is implemented. The **-control_signal** option specifies if a scan enable or a test mode signal is used to drive the OR gate. The test mode signal is held constant during testing. The scan enable signal is guaranteed to be logic 1 only during scan-in and scan-out. A port is created in the design according to the chosen test control signal.

Second, due to the control point OR gate, the enable signal is not observable if a test mode signal has been chosen as input to the control point OR gate. To improve the observability, the **-observation_point** option enables the insertion of observation points on the enable signals. The enable signals are grouped according to the clock signal and the clock edge of the register with which

they are associated. An XOR gate tree fed by enable signals and providing input to an observability register is implemented for each group. The maximum depth of the XOR tree is determined by the **-observation_logic_depth** option. To minimize power consumption in the observability circuitry, the enable signals and the observability register's clock signal are gated by the test control signal.

The **-max_fanout** option enables you to specify a number for the maximum fanout of clock-gating cell at the RTL level. Based on the `max_fanout_count`, the Power Compiler tool splits the register banks that are to be clock-gated, and creates as many clock-gating cells that is required to drive them.

This option does not override the maximum fanout design rule constraint during the `compile_ultra` command. The value specified with the **-max_fanout** option is an integer, greater than 0. It should not be less than the `minimum_bitwidth` value. The default is a huge number (unlimited for all practical purposes).

The **-num_stages** option enables multistage clock gating with the **compile_ultra -gate_clock** command. The `num_stages_count` value is an integer greater than 0. It determines the maximum number of stages that are introduced by factoring of the inserted clock gates. The default is 1, so no factoring is performed. The number of stages are counted from the register to the clock source. Any multiple-input combinational cell in the clock path will stop the stage count.

The **-no_sharing** option can be used only with the **-max_fanout** option. When you use this option, Power Compiler does not share any clock-gating cell between register banks. When you do not use this option, Power Compiler enables you to share clock-gating cells between register banks with similar control logic to meet the `max_fanout_count` value.

If **compile_ultra -gate_clock** or **replace_clock_gates** is issued without explicitly setting the clock-gating style beforehand (such as when the **set_clock_gating_style** command is not issued) a default style is applied.

The default style is derived from the *target_library* that is set when the **compile_ultra -gate_clock** or **replace_clock_gates** command is issued. The style is chosen such that the best available clock gating configuration is used with the current *target_library*. The "best available" style corresponds to the first legal style in context of the current *target_library* of the following combinations:

- (a) **set_clock_gating_style -pos integrated -neg integrated **
-control_point before -control_signal scan_enable
- (b) **set_clock_gating_style -pos integrated -neg integrated **
-control_point after -control_signal scan_enable
- (c) **set_clock_gating_style -pos integrated -neg integrated **
**-control_point before -control_signal test_mode **
-observation_point true
- (d) **set_clock_gating_style -pos integrated -neg integrated **
**-control_point after -control_signal test_mode **
-observation_point true
- (e) **set_clock_gating_style -pos integrated -neg integrated**
- (f) **set_clock_gating_style -pos integrated -neg or **
-control_point before -control_signal scan_enable
- (g) **set_clock_gating_style -pos integrated -neg or **
-control_point after -control_signal scan_enable
- (h) **set_clock_gating_style -pos integrated -neg or **
**-control_point before -control_signal test_mode **
-observation_point true
- (i) **set_clock_gating_style -pos integrated -neg or **
**-control_point after -control_signal test_mode **

-observation_point true

(j) **set_clock_gating_style -pos integrated -neg or**

(k) **set_clock_gating_style -pos and -neg integrated **
-control_point before -control_signal scan_enable

(l) **set_clock_gating_style -pos and -neg integrated **
-control_point after -control_signal scan_enable

(m) **set_clock_gating_style -pos and -neg integrated **
**-control_point before -control_signal test_mode **
-observation_point true

(n) **set_clock_gating_style -pos and -neg integrated **
**-control_point after -control_signal test_mode **
-observation_point true

(o) **set_clock_gating_style -pos and -neg integrated**

(p) **set_clock_gating_style -pos and -neg or**

Instance specific Clock-gating Style

Instance specific clock-gating style refers to the possibility of setting a clock gating style on a specific instances of the design. The **-instances**, **-power_domains** and **-designs** options allow you to specify the set of hierarchical cells, power domains and designs on which a given clock gating style should be applied.

To enable instance specific clock gating style, set the **power_cg_iscgs_enable** variable to **true**.

When instance specific clock gating style is disabled, the use of the **-instances**, **-power_domains** and **-designs** options is not allowed, and **set_clock_gating_style** command errors out. If **-instances**, **-power_domains** and **-designs** are not used, the clock gating style is associated with the current design.

The following method is used to set the style on a hierarchical instance:

1. Look for a style set on that instance.
2. If no style has been found and the instance was used to define a power_domain, try to find a style set on the power_domain.
3. If no style has been found, look for a style set on the design it is referencing.
4. If no style has been found, it inherits the style from the hierarchical cell containing it.

Note that instances that have a specific clock gating style set are not automatically ungrouped.

EXAMPLES

Default values are assumed for all unspecified options. In particular, option values from previous clock-gating style settings are not stored.

The following example sets the clock-gating style such that banks with two or more flip-flops are implemented with gated clocks:

```
prompt> set_clock_gating_style -minimum_bitwidth 2
```

The following example specifies the clock-gating circuitry, including the library cells for gating the clock of registers inferred by a positive edge clock HDL code construct. Note that for the negative-edge registers, the default is used. The latch-based style is chosen, therefore, the circuitry must have AND functionality. A particular latch LD2 is chosen from the target library.

```
prompt> set_clock_gating_style -sequential_cell latch:LD2 \
    -positive_edge_logic {nand:NAND2 inv:IV3}
```

The following example specifies the clock-gating circuitry for registers inferred by positive and negative edge clock HDL code constructs. By default, latch-based clock gating is assumed. Three gates are specified on the clock network. The mapping to library cells happens automatically during compilation. A hold time constraint of 0.5 is specified. By default, the setup time constraint is 0.

```
prompt> set_clock_gating_style -pos {buf nand inv} \
    -neg {buf or inv} -hold 0.5
```

The following example specifies the clock-gating circuitry for registers inferred by positive and negative edge clock HDL code constructs (latch-free clock gating). Note that an inverter is inferred for those registers that are inferred by a negative-edge clock construct to obtain an inverted clock signal at the flip-flops clock pin.

```
prompt> set_clock_gating_style -sequential_cell none \
    -pos {or} -neg {and}
```

The following example specifies that clock gating is performed with the insertion of control point OR gates:

```
prompt> set_clock_gating_style -control_point after \
    -control_signal scan_enable
```

The following example specifies that clock gating is performed with the insertion of control point OR gates, as well as the insertion of observation points:

```
prompt> set_clock_gating_style -control_point before \
    -control_signal test_mode -observation_point true \
    -observation_logic_depth 4
```

The following example specifies a latch based clock-gating style and limits the fanout of individual clock-gating cells to 8. The **no_sharing** option ensures that a single clock-gating cell gates registers from only one register bank.

```
prompt> set_clock_gating_style -sequential latch \
    -max_fanout 8 -no_sharing
```

The following example specifies the clock-gating circuitry for registers inferred by positive and negative edge clock HDL code constructs. Note that an active low enabled and inverting gclk output integrated clock-gating cell is chosen for registers that are inferred by a positive-edge clock construct. An inverting gclk output integrated clock-gating cell is chosen for registers that are inferred by a negative-edge clock construct.

```
prompt> set_clock_gating_style \
    -pos {integrated active_low_enable invert_gclk} \
    -neg {integrated invert_gclk}
```

The following example specifies two different latches for positive and negative-edge logic:

```
prompt> set_clock_gating_style \
    -positive_edge_logic {latch:lsi_10k/LD2 and} \
    -negative_edge_logic {latch:lsi_10k/LD1 or}
```

The following example specifies a clock gating style with minimum bitwidth and maximum fanout of 1 on instances m1/b1 and m2:

```
prompt> set_clock_gating_style -minimum_bitwidth 1 \
    -max_fanout 1 -instances {m1/b1 m2}
```

The following example specifies a clock-gating style with integrated cells on instances m1 and power domain PD0:

```
prompt> set_clock_gating_style -positive_edge_logic {integrated} -instances m1 \  
       -power_domains PD0
```

The following example specifies a clock-gating style with integrated cells set on instances m1, power domain PD0 and all instances of design adder16:

```
prompt> set_clock_gating_style -positive_edge_logic {integrated} -instances m1 \  
       -power_domains PD0 -designs adder16
```

SEE ALSO

- compile(2)
- compile_ultra(2)
- insert_dft(2)
- propagate_constraints(2)
- remove_clock_gating(2)
- replace_clock_gates(2)
- set_clock_gating_objects(2)
- power_cg_derive_related_clock(3)
- power_cg_ignore_setup_condition(3)
- power_cg_permit_opposite_edge_icg(3)

set_clock_groups

Specifies clock groups that are mutually exclusive or asynchronous with each other in a design so that the paths between these clocks are not considered during timing analysis.

SYNTAX

```
status set_clock_groups
  -physically_exclusive
  | -logically_exclusive
  | -asynchronous [-allow_paths]
  [-name clock_group_name]
  -group clock_list
  [-comment comment_string]
```

Data Types

```
clock_group_name  string
clock_list        collection
comment_string    string
```

ARGUMENTS

-physically_exclusive

Specifies that the clock groups are physically exclusive with each other. Physically-exclusive clocks cannot coexist in the design physically. An example of this is multiple clocks that are defined on the same source pin. The **-physically_exclusive**, **-logically_exclusive**, and **-asynchronous** options are mutually exclusive; you must choose only one.

-logically_exclusive

Specifies that the clock groups are logically exclusive with each other. An example of logically-exclusive clocks is multiple clocks that are selected by a multiplexer but might have coupling with each other in the design. However, you should not set these multiplexed clocks to be exclusive if some physical paths exist among them somewhere in the design. The **-physically_exclusive**, **-logically_exclusive**, and **-asynchronous** options are mutually exclusive; you must choose only one.

-asynchronous

Specifies that the clock groups are asynchronous to each other. Two clocks are asynchronous with respect to each other if they have no phase relationship at all. Signal integrity analysis uses an infinite arrival window on the aggressor unless all the arrival windows on the victim net and the aggressor net are defined by synchronous clocks. The **-physically_exclusive**, **-logically_exclusive**, and **-asynchronous** options are mutually exclusive; you must choose only one.

-allow_paths

Enables timing analysis between the specified clock groups. If this option is not specified, timing analysis among the defined

clock groups are disabled. This option can be used with only with asynchronous clock groups.

-name *clock_group_name*

Specifies a name for the clock grouping. Each command must specify a unique name that identifies the exclusive or asynchronous relationship among the specified clock groups. This name is used later to easily remove the clock grouping defined here. By default, the command creates a unique name.

-group *clock_list*

Specifies the clocks in a group. You can use the **-group** option more than once in a single command execution. However, you can include a clock only in one group in a single command execution. To include a clock in multiple groups, you must execute the **set_clock_groups** command multiple times.

By default, a generated clock and its master clock are not in the same group when the exclusive or asynchronous clock groups are defined. Put these clocks explicitly in the same group if needed.

Each **-group** instance specifies a group of clocks, which are exclusive or asynchronous with the clocks in all other groups. If you specify only one group, it means that the clocks in that group are exclusive or asynchronous with all other clocks in the design. A default other group is created for this single group. Whenever a new clock is created, it is automatically included in this group.

-comment *comment_string*

Allows the command to accept a comment string. The tool honors the annotation and preserves it with the SDC object so that the exact string is written out when the constraint is written out when you use the **write_sdc** or **write_script** command. The comment remains intact through the synthesis, place-and-route, and timing-analysis flows.

DESCRIPTION

This command specifies clock groups that are mutually exclusive, either physically or logically, or asynchronous with each other in a design.

If multiple clock group relationships are defined for the same pair of clocks, physically exclusive has the highest precedence, followed by asynchronous, and logically exclusive has the lowest precedence.

By default, the timing paths between these clocks are not considered during timing analysis. This is similar to declaring false paths between these clocks. Thus, you do not have to manually set a false path if you have already specified two clocks as exclusive or asynchronous. If a false path is already set between two clocks when you specify that they are exclusive or asynchronous, the false path is overwritten by the **set_clock_groups** command. Other exceptions are unaffected.

Note that for asynchronous clock groups, you can consider the timing paths between these clocks by using the **-allow_paths** option.

The two types of exclusive clocks are not treated differently for simple timing analysis. However, signal integrity analysis treats logically-exclusive clocks as synchronous for timing windows. Physically-exclusive clocks are not considered for timing window analysis with each other.

When the clocks are asynchronous to each other, crosstalk analysis ignores the timing relationship between them during the timing window overlap analysis. This case is also referred to as infinite window overlap. The results can be optimistic without infinite window overlap for the synchronous clocks. It is important to use the **set_clock_groups -asynchronous** command when the clocks are not synchronous to each other.

To undo the **set_clock_groups** command, use the **remove_clock_groups** command. To report the clock groups defined in a design, use the **report_clock** command with the **-groups** option.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example defines two asynchronous clock domains:

```
prompt> set_clock_groups -asynchronous -name g1 \  
-group CLK1 -group CLK2
```

The following example defines a clock named CLK1 as asynchronous with all other clocks in the design:

```
prompt> set_clock_groups -asynchronous -group CLK1
```

The following example shows how to simultaneously analyze multiple clocks per register without manually specifying false paths.

Assume that a design contains two pairs of mutually-exclusive clocks that are multiplexed: CLK1 and CLK2, and CLK3 and CLK4.

If each pair of clocks is selected by a different signal, you must execute two **set_clock_groups** commands to simultaneously analyze all four clocks:

```
prompt> set_clock_groups -logically_exclusive \  
-group CLK1 -group CLK2  
prompt> set_clock_groups -logically_exclusive \  
-group CLK3 -group CLK4
```

If each pair of clocks is selected by the same signal, you must execute only one **set_clock_groups** command to simultaneously analyze all four clocks:

```
prompt> set_clock_groups -logically_exclusive \  
-group {CLK1 CLK3} -group {CLK2 CLK4}
```

The following example shows how to define clocks CLK1 and CLK2 as physically exclusive:

```
prompt> set_clock_groups -physically_exclusive \  
-group {CLK1} -group {CLK2}
```

SEE ALSO

```
create_clock(2)  
create_generated_clock(2)  
remove_clock_groups(2)  
report_clock(2)  
set_false_path(2)
```

set_clock_latency

Specifies the clock latency for clocks, ports, or pins.

SYNTAX

```
status set_clock_latency
[-rise]
[-fall]
[-min]
[-max]
[-source]
[-early]
[-late]
[-dynamic jitter]
[-clock clock_list]
delay
object_list
```

Data Types

```
jitter      float
clock_list  list
delay       float
object_list collection
```

ARGUMENTS

-rise

Applies the specified latency value only to rising transitions at register clock pins. By default, the latency value applies to both rising and falling transitions at register clock pins.

-fall

Applies the specified latency value only to falling transitions at register clock pins.

-min

Applies the specified latency value only to the minimum operating condition. By default, the latency value applies to both the minimum and maximum operating conditions.

-max

Applies the specified latency value only to the maximum operating condition.

-source

Specifies the source (instead of network) latency. Source latency is the amount of delay from the ideal waveform to the source object in the design, as defined by the **create_clock** command.

By default, the delay value applies to the network latency, the amount of delay from the source object in the design to the register clock pin of the sequential device. The total latency is source latency plus network latency.

-early

Applies the specified delay value only to early source latency, for example, to calculate the clock delay to the capture register in a setup check. This option can be used only with the **-source** option. By default, when you use the **-source** option, the specified delay value applies to both early and late source latency.

-late

Applies the specified delay value only to late source latency, for example, to calculate the clock delay to the capture register in a hold check. This option can be used only with the **-source** option.

-dynamic jitter

Specifies the dynamic component of the clock latency, which represents the amount of jitter in the original clock source. This option can be used only with the **-source** option.

-clock *clock_list*

Applies the latency value with respect to the specified clock. This option lets you specify the relevant clock when you set the latency on a port or pin, and multiple clocks pass through the port or pin.

delay

Specifies the clock latency value, either network latency by default, or source latency if you use the **-source** option.

object_list

Specifies the clocks, ports, and pins on which to set the delay value.

DESCRIPTION

This command specifies clock latency, which is the amount of delay for a clock signal reaching the clock pin of a sequential device. You can specify two types of clock latency: network latency (the default) and source latency (by using the **-source** option).

Clock network latency is the time it takes a clock signal to propagate from the clock definition point (as defined by the **create_clock** command) to a register clock pin. The rise and fall latencies are the latencies for rising and falling transitions at the register clock pin, respectively. The tool considers inversion of the clock waveform, if present in the clock network, to determine the rising or falling sense at the register clock pin.

Clock source latency (also called insertion delay) is the time it takes for a clock signal to propagate from its actual ideal waveform origin point to the clock definition point in the design. You can use it to model off-chip clock latency when a clock generation circuit is not part of the current design. For generated clocks, you can use clock source latency to model the delay from the master clock to the generated clock definition point. You can use the **-early** and **-late** options to specify early and late clock source latencies, respectively.

The dynamic component of clock latency is the amount of jitter in the original clock source, which leads to a reduction in timing slack equal to the specified amount of time. You specify this amount by using the **-dynamic** option, which can be used only with the **-source** option. In the case of a zero-cycle path, in which the same clock edge both launches and captures data in the path, jitter does not reduce the slack; the tool takes this into account in any clock reconvergence pessimism removal adjustment performed on the path.

If multiple clocks are allowed per object, you can specify the clock latency with respect to specific clocks by using the **-clock** option. By using multiple **set_clock_latency** commands, you can specify different latency values for different clocks that pass through the same ports or pins.

By default, the tool assumes ideal clocking, which means clocks have a specified network latency (from the **set_clock_latency** command) or zero network latency by default. Propagated clock network latency (from the **set_propagated_clock** command) is normally used after layout and final clock tree generation. Specifying the ideal clock network latency provides an estimate of the clock tree delay for before layout.

You can specify clock source latency using the **-source** option for both ideal and propagated clocks. The total clock latency at a register clock pin is the sum of the source latency and network latency.

When you apply the **set_clock_latency** command to pins or ports, it affects all register clock pins in the transitive fanout of the specified pins or ports. You can apply source latency only to clocks and clock source pins.

To undo the effects of the **set_clock_latency** command, use the **remove_clock_latency** command.

To report the clock network latency and clock source latency information, use the **report_clock -skew** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example specifies a rise latency of 1.2 and a fall latency of 0.9 for the clock named CLK1:

```
prompt> set_clock_latency 1.2 -rise [get_clocks CLK1]
prompt> set_clock_latency 0.9 -fall [get_clocks CLK1]
```

The next example specifies an early rise and an early fall source latency of 0.8, and a late rise and a late fall source latency of 0.9 for the clock named CLK1:

```
prompt> set_clock_latency 0.8 -source -early [get_clocks CLK1]
prompt> set_clock_latency 0.9 -source -late [get_clocks CLK1]
```

The next example specifies an early and late source latency of 3 and 5 respectively with dynamic components of 0.5.

```
prompt> set_clock_latency 3 -source -early -dynamic 0.5 [get_clocks CLK1]
prompt> set_clock_latency 5 -source -late -dynamic 0.5 [get_clocks CLK1]
```

SEE ALSO

- create_clock(2)
- current_design(2)
- remove_clock_latency(2)
- report_clock(2)
- set_clock_gate_latency(2)
- set_clock_transition(2)
- set_clock_uncertainty(2)
- set_input_delay(2)
- set_propagated_clock(2)

set_clock_sense

Specifies the clock sense (with respect to the clock source) propagating forward from the specified pins.

SYNTAX

```
status set_clock_sense  
-stop_propagation  
| -logical_stop_propagation  
| -positive  
| -negative  
| -pulse pulse_type  
[-clocks clocks]  
pins
```

Data Types

```
pulse_type  string  
clocks      collection  
pins        collection
```

ARGUMENTS

-stop_propagation

Stops propagation of the specified clocks at the specified pins along the clock paths only.

The **-stop_propagation** option is mutually exclusive with the **-positive**, **-negative**, **-logical_stop_propagation**, and **-pulse** options. You must specify one of these options.

-logical_stop_propagation

Stops propagation of the specified clocks at the specified pins along the clock paths only.

The **-logical_stop_propagation** option is mutually exclusive with the **-positive**, **-negative**, **-stop_propagation**, and **-pulse** options. You must specify one of these options.

-positive

Propagates only the positive unate paths forward from the specified pins.

The **-positive** option is mutually exclusive with the **-negative**, **-pulse**, **-logical_stop_propagation**, and **-stop_propagation** options. You must specify one of these options.

-negative

Propagates only the negative unate paths forward from the specified pins.

The **-negative** option is mutually exclusive with the **-positive**, **-pulse**, **-logical_stop_propagation**, and **-stop_propagation** options. You must specify one of these options.

-pulse *pulse_type*

Specifies the type of pulse clock that is generated from the specified pins. You must specify one of the following values: **rise_triggered_high_pulse**, **rise_triggered_low_pulse**, **fall_triggered_high_pulse**, or **fall_triggered_low_pulse**.

The **-pulse** option is mutually exclusive with the **-positive**, **-negative**, **-logical_stop_propagation**, and **-stop_propagation** options. You must specify one of these options.

-clocks *clocks*

Specifies the clocks to which the specified clock sense applies.

If you do not specify this option, the clock sense setting applies to any clock that passes through the specified pins.

pins

Specifies the pins on which to set the clock sense.

DESCRIPTION

This command provides the capability to define the clock sense at nonunate points in the clock network. The command is ignored if it is used on a unate point in the clock network. The specified clock sense propagates forward from the specified pins.

If the **-clocks** option is used, only the specified clocks are considered. Otherwise, all clocks passing through the specified pins are considered.

Warning messages are issued if the specified sense cannot be respected on the specified pins. Hierarchical pins are not supported.

To undo the settings made by the **set_clock_sense** command, use the **remove_clock_sense** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example specifies positive unateness for a pin named XOR/Z with respect to the CLK1 clock:

```
prompt> set_clock_sense -positive -clocks [get_clocks CLK1] XOR/Z
```

The following example specifies a pulse type of **rise_triggered_high_pulse** for all clocks passing through a pin named MUX/Z:

```
prompt> set_clock_sense -pulse rise_triggered_high_pulse MUX/Z
```

SEE ALSO

`remove_clock_sense(2)`

set_clock_transition

Sets the transition time at the clock pins of all sequential devices clocked by the specified ideal clocks.

SYNTAX

```
status set_clock_transition  
    transition  
    [-rise | -fall]  
    [-min]  
    [-max]  
    clock_list
```

Data Types

```
transition    float  
clock_list    collection
```

ARGUMENTS

transition

Specifies the transition time at the clock pins of the sequential devices clocked by the specified ideal clocks, in time units of the technology library used during optimization.

-rise | -fall

Specifies whether the value applies to rising or falling transitions at the clock pins of the sequential devices. If neither is specified, both rising and falling clock transitions are set. If you specify only **-rise** or only **-fall** and do not specify a transition time for the other type of transition, the default value, 0.0, applies to the other type of transition.

-min

Specifies that the value applies to minimum delay analysis only. By default, the transition value applies to both minimum and maximum delay analysis.

-max

Specifies that the value applies to maximum delay analysis only. By default, the transition value applies to both minimum and maximum delay analysis.

clock_list

Specifies the ideal clocks to which the transition value applies. The specified transition time applies to all register clock pins in the transitive fanout of each specified clock.

DESCRIPTION

This command explicitly specifies the transition time of clock signals that reach the clock pins of sequential devices in the transitive fanout of one or more ideal clocks.

This command is especially useful before layout, when clock trees are incomplete and the clocks have ideal behavior. By default, ideal clocks have a transition time of zero. To get more accurate timing results, specify an estimated nonzero transition time.

If you specify the name of a propagated clock instead of an ideal clock, the transition time value is ignored; the calculated transition time is used instead. You should set clocks to be propagated clocks only after the final clock trees are constructed.

The command sets one or more of the **clock_transition_fall_max**, **clock_transition_fall_min**, **clock_transition_rise_max**, and **clock_transition_rise_min** attributes of the specified clock objects.

To undo the effects of the **set_clock_transition** command, use the **remove_clock_transition**, **remove_attribute**, or **reset_design** command.

To list all clock transition values that have been set, use the **report_clock -skew** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following command sets both the rise and fall transition times to 0.75 on the clock pins of all sequential elements clocked by the clock named CLK1:

```
prompt> set_clock_transition 0.75 CLK1
```

The following command sets the fall transition time to 0.64 for maximum-delay analysis on the clock pins of all sequential elements clocked by CLK1 and CLK2:

```
prompt> set_clock_transition 0.64 -fall -max {CLK1 CLK2}
```

The following command sets both the rise and fall transition times to 0.20 on the clock pins of all sequential elements clock by ideal clocks:

```
prompt> set_clock_transition 0.20 [all_clocks]
```

SEE ALSO

- all_clocks(2)
- create_clock(2)
- get_clocks(2)
- remove_attribute(2)
- report_clock(2)
- remove_clock_transition(2)
- reset_design(2)

set_clock_uncertainty

Specifies the uncertainty (skew) of the specified clock networks.

SYNTAX

```
status set_clock_uncertainty
  [object_list

    | -from from_clock
    | -rise_from rise_from_clock
    | -fall_from fall_from_clock
    -to to_clock
    | -rise_to rise_to_clock
    | -fall_to fall_to_clock]
  [-rise]
  [-fall]
  [-setup]
  [-hold]
  uncertainty
```

Data Types

<i>object_list</i>	collection
<i>from_clock</i>	collection
<i>rise_from_clock</i>	collection
<i>fall_from_clock</i>	collection
<i>to_clock</i>	collection
<i>rise_to_clock</i>	collection
<i>fall_to_clock</i>	collection
<i>uncertainty</i>	float

ARGUMENTS

object_list

Specifies the clocks, ports, or pins for simple uncertainty. The uncertainty is applied either to the capturing latches clocked by one of the specified clocks or to the capturing latches whose clock pins are in the fanout of a specified port or pin.

You must specify either this argument or a clock pair with the **-from/-rise_from/-fall_from** and **-to/-rise_to/-fall_to** options; you cannot specify both.

-from *from_clock*

Specifies the source clocks for interclock uncertainty.

You must specify either a clock pair with the **-from/-rise_from/-fall_from** and **-to/-rise_to/-fall_to** options or the *object_list* argument; you cannot specify both.

-rise_from rise_from_clock

Specifies that the uncertainty applies only to the rising edge of the source clock.

You can use only one of the **-from**, **-rise_from**, and **-fall_from** options.

-fall_from fall_from_clock

Specifies that the uncertainty applies only to the falling edge of the source clock.

You can use only one of the **-from**, **-rise_from**, and **-fall_from** options.

-to to_clock

Specifies the destination clocks for interclock uncertainty.

You must specify either a clock pair with the **-from/-rise_from/-fall_from** and **-to/-rise_to/-fall_to** options or the *object_list* argument; you cannot specify both.

-rise_to rise_to_clock

Specifies that the uncertainty applies only to the rising edge of the destination clock.

You can use only one of the **-to**, **-rise_to**, and **-fall_to** options.

-fall_to fall_to_clock

Specifies that the uncertainty applies only to the falling edge of the destination clock.

You can use only one of the **-to**, **-rise_to**, and **-fall_to** options.

-rise

Indicates that the uncertainty applies to only the rising edge of the destination clock.

By default, the uncertainty applies to both the rising and falling edges.

This option is valid only for interclock uncertainty, and is now obsolete. Unless you need this option for backward-compatibility, use the **-rise_to** option instead.

-fall

Indicates that the uncertainty applies to only the falling edge of the destination clock.

By default, the uncertainty applies to both rising and falling edges.

This option is valid only for interclock uncertainty, and is now obsolete. Unless you need this option for backward-compatibility, use the **-fall_to** option instead.

-setup

Indicates that the uncertainty applies only to setup checks.

By default, the uncertainty applies to both setup and hold checks.

-hold

Indicates that the uncertainty applies only to hold checks.

By default, the uncertainty applies to both setup and hold checks.

uncertainty

Specifies a floating-point number that indicates the uncertainty value. Typically, clock uncertainty should be positive. Negative uncertainty values are supported for constraining designs with complex clock relationships. Setting the uncertainty value to a negative number could lead to optimistic timing analysis and should be used with extreme care.

DESCRIPTION

This command specifies the clock uncertainty (skew characteristics) of the specified clock networks. This command can specify either interclock uncertainty or simple uncertainty. For interclock uncertainty, use the **-from/-rise_from/-fall_from** and **-to/-rise_to/-fall_to** options to specify the source clock and the destination clock; all paths between these clocks receive the uncertainty value. For simple uncertainty, use the *object_list* argument; the uncertainty value is either to the capturing latches clocked by one of the clocks in the *object_list* argument or to the capturing latches whose clock pins are in the fanout of a port or pin specified in the *object_list* argument.

Set the uncertainty to the worst skew expected to the endpoint or between the clock domains. You can increase the value to account for additional margin for setup and hold.

When you specify interclock uncertainty, ensure that you specify it for all possible interactions of the clock domains. For example, if you specify paths from CLKA to CLKB and CLKB to CLKA, you must specify the uncertainty for both even if the values are the same.

Interlock uncertainty is more specific than simple uncertainty. If a command that specifies interlock uncertainty conflicts with a command that specifies simple uncertainty, the command that specifies interlock uncertainty takes precedence.

If there is no applicable interlock uncertainty for a path, the value for simple uncertainty is used.

To remove the uncertainties set by the **set_clock_uncertainty** command, use the **remove_clock_uncertainty** command.

To view clock uncertainty information, use the **report_clock -skew** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example specifies that all paths leading to registers or ports clocked by CLK have setup uncertainty of 0.65 and hold uncertainty of 0.45:

```
prompt> set_clock_uncertainty -setup 0.65 [get_clocks CLK]
prompt> set_clock_uncertainty -hold 0.45 [get_clocks CLK]
```

The following example specifies interclock uncertainties between the PHI1 and PHI2 clock domains:

```
prompt> set_clock_uncertainty 0.4 -from PHI1 -to PHI1
prompt> set_clock_uncertainty 0.4 -from PHI2 -to PHI2
prompt> set_clock_uncertainty 1.1 -from PHI1 -to PHI2
prompt> set_clock_uncertainty 1.1 -from PHI2 -to PHI1
```

The following example specifies interclock uncertainties between the PHI1 and PHI2 clock domains with specific edges:

```
prompt> set_clock_uncertainty 0.4 -rise_from PHI1 -to PHI2
```

```
prompt> set_clock_uncertainty 0.4 -fall_from PHI2 -rise_to PHI2  
prompt> set_clock_uncertainty 1.1 -from PHI1 -fall_to PHI2
```

The following example shows conflicting **set_clock_uncertainty** commands, one for simple uncertainty and one for interclock uncertainty. The interclock uncertainty value of 2 takes precedence.

```
prompt> set_clock_uncertainty 5 [get_clocks CLKA]  
prompt> set_clock_uncertainty 2 \  
       -from [get_clocks CLKB] -to [get_clocks CLKA]
```

The following example specifies the uncertainty from CLKA to CLKB and from CLKB to CLKA. Notice that both must be specified even though the value is the same.

```
prompt> set_clock_uncertainty 2 \  
       -from [get_clocks CLKA] -to [get_clocks CLKB]  
prompt> set_clock_uncertainty 2 \  
       -from [get_clocks CLKB] -to [get_clocks CLKA]
```

The following example illustrates a situation in which simple uncertainty is used when there is no applicable interclock uncertainty for a path. The first command specifies a simple uncertainty of 5 for CLKA paths, and the second command specifies an interclock uncertainty of 2 for paths from CLKB to CLKA. If there are paths between CLKA and other clocks (for example, CLKC, CLKD, and so on) for which interclock uncertainty has not been specifically defined, the simple uncertainty (in this case, 5) is used.

```
prompt> set_clock_uncertainty 5 [get_clocks CLKA]  
  
prompt> set_clock_uncertainty 2 \  
       -from [get_clocks CLKB] -to [get_clocks CLKA]
```

SEE ALSO

```
remove_clock_uncertainty(2)  
report_clock(2)  
set_clock_latency(2)  
set_clock_transition(2)
```

set_combinational_type

Sets attributes on cell instances to specify which combinational cells from the target library are to be used by the **compile** command.

SYNTAX

```
status set_combinational_type  
-replacement_gate replacement_gate  
[cell_list]
```

Data Types

```
replacement_gate  string  
cell_list         list
```

ARGUMENTS

-replacement_gate *replacement_gate*

Specifies a combinational gate from the target library to use by the **compile** command as the exact replacement gate for the cells in the cell list. This option sets the **combinational_type_exact** attribute to the *replacement_gate* on all cells in *cell_list*.

cell_list

Specifies a list of cells in which to use the replacement combinational gate.

DESCRIPTION

The **set_combinational_type** command specifies replacement combinational gate information for the **compile** command to use by setting attributes on the cell instances. Specifying **-replacement_gate** sets the **combinational_type_exact** attribute to *replacement_gate*, indicating to use this attribute as the replacement gate for those cells.

In the mapping process for combinational gates, **compile** attempts to convert combinational gates tagged by **set_combinational_type** to the specified replacement combinational gate. In the absence of attributes on a combinational cell instance to control the mapping, **compile** chooses the one best for timing, power, or area.

After mapping the cell instances to the replacement gate, **compile** sets the **dont_touch** attribute on these cell instances to disable further optimization on these cells.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example indicates that the replacement combinational gate for cell *U1* is cell *AN2P*. This sets the **combinational_type_exact** attribute to *AN2P* on the cell *U1*. The presence of this attribute indicates that cell *U1* attempts an exact mapping to gate *AN2P*.

```
prompt> set_combinational_type -replacement_gate AN2P U1
```

SEE ALSO

- current_design(2)
- get_attribute(2)
- remove_attribute(2)
- reset_design(2)
- translate(2)
- attributes(3)
- target_library(3)

set_compile_directives

Controls the application of high-level optimization operations on cells, hierarchical pins, references, designs, and library cells.

SYNTAX

```
status set_compile_directives
      object_list
      [-delete_unloaded_gate true | false]
      [-constant_propagation true | false]
      [-local_optimization true | false]
      [-critical_path_resynthesis true | false]
```

ARGUMENTS

object_list

Applies the **set_compile_directive** command on the list of objects (cells, hierarchical pins, references, designs, and library cells) you specify. This command is not applied on classes of cells that can only be internally generated (for example, generic logic, synthetic operators, and control logic to selector operators). If you include more than one object, the objects must be enclosed in quotation marks (") or braces ({}). For more information about object names, see the **get_object_name** command man page.

-delete_unloaded_gate true | false

Allows you to delete unloaded gates from the cells specified in the *object_list* when set to **true** (the default). When set to **false**, cells referenced by *object_list* are not deleted, even if they do not drive any load.

-constant_propagation true | false

Allows constants to be propagated through the cells or hierarchical pins specified in *object_list* when set to **true** (the default). When set to **false**, constant propagation, within or across hierarchies, is disabled for the specified cells or hierarchical pins.

-local_optimization true | false

Allows you to perform operations on the cells specified in the *object_list* when set to **true** (the default). When set to **false**, operations that involve the immediate neighborhood of cells referenced by the object list are not performed. These operations include phase shift operations.

-critical_path_resynthesis true | false

Allows you to perform critical path resynthesis on cells specified in the *object_list* when set to **true** (the default). When set to **false**, critical path resynthesis is not performed on cells referenced by *object_list*.

DESCRIPTION

You must have a DC Ultra license to use this command.

The **set_compile_directives** command controls switching on or off of certain high-level optimization steps on cells, hierarchical pins, references, designs, and library cells, where hierarchical pins are only accepted for the **-constant_propagation** option. The high-level optimization steps controlled by this command include deletion of unloaded gates, propagation of constants, local optimization (buffering, phase operations, and so on), and critical path resynthesis. Note that changing the default value of **true** of any of these command-line options also implies that the cells referenced by the object list are not structured.

An important application of this command is to direct synthesis to perform only sizing operations on a cell. You can achieve this by setting all command options to **false**. The **set_size_only** command achieves the same result. The **set_size_only** command does not require a DC Ultra license.

Setting **set_compile_directives** on a hierarchical cell implies **set_compile_directives** on all cells below it. Setting **set_compile_directives** on a library cell implies **set_compile_directives** on all instances of that cell. Setting **set_compile_directives** on a reference implies **set_compile_directives** on all cells of that reference during subsequent optimizations of the design.

Setting the compile directives on a design has an effect only when the design is instantiated within another design as a level of hierarchy. In this case, **set_compile_directives** on the design implies that all cells under that level of hierarchy acquire the same directives. Setting **set_compile_directives** on the top-level design has no effect because the top-level design is not instantiated within any other design.

Note that synthetic parts and generic cells ignore the **set_compile_directives** command; for example, many of the cells read in from an HDL description. Control logic feeding selector operator cells also ignore this command. Thus, the command is not applied on any class of cells that can only be internally generated.

When you specify an object name, **set_compile_directives** first looks within the current design for a hierarchical pin that matches that name. If it finds a match, the compile directives are assigned to that hierarchical pin. If no match is found, the command then looks for a cell that matches that name. If it finds a match, the compile directives are assigned to that cell. If no match is found, the command then looks for a reference, then a design in the system, and finally for a library cell in the system. This command assigns compile directives to the first object for which it finds a match.

The **report_cell** command prints all the compile directives set by the command in the attribute column.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command specifies to size the *block1* and *analog1* cells during **compile** only:

```
prompt> set_compile_directives -del false -const false -local false \
        -crit false [get_cells {block1 analog1}]
```

SEE ALSO

compile(2)
get_object_name(2)
report_cell(2)
set_size_only(2)

set_compile_partitions

Specifies the compile partitions for the current design.

SYNTAX

```
status set_compile_partitions  
  -level level  
  | -designs design_list  
  | -all  
  | -auto  
  [-force]  
  [-no_reset]
```

Data Types

level integer
design_list list

ARGUMENTS

-level *level*

Creates partitions from all designs at the specified hierarchical level. Level one selects the subdesigns of the top-level design. The specified level must be greater than or equal to one. You must specify one, but not more than one, of the following options: **-level**, **-designs**, **-all**, or **-auto**.

-designs *design_list*

Creates partitions from the specified designs. You must specify one, but not more than one, of the following options: **-level**, **-designs**, **-all**, or **-auto**.

-all

Creates partitions from all subdesigns throughout the hierarchy of the current design. You must specify one, but not more than one, of the following options: **-level**, **-designs**, **-all**, or **-auto**.

-auto

Creates partitions from all designs automatically. You must provide the threshold for the partition size. See the *Automated Chip Synthesis User Guide* for threshold setting. You must specify one, but not more than one, of the following options: **-level**, **-designs**, **-all**, or **-auto**.

-force

Enables partitioning, even if the design contains multiple instances. If the specified design contains multiple instances, Automated Chip Synthesis selects as the master instance the instance with the hierarchical name that is first alphabetically. For

more information about master instances, see the **MasterInstance** attribute man page.

By default, the **set_compile_partitions** command fails if the design contains multiple instances. You must either resolve these instances before running the command, or use the **-force** option.

-no_reset

Adds the specified partitions to the existing partition definitions.

By default, the **is_partition** attributes are removed from all subdesigns before creating the specified partitions.

DESCRIPTION

The **set_compile_partitions** command creates partitions from the specified designs by setting the **is_partition** attribute on the designs. When Automated Chip Synthesis (ACS) compiles a top-level design, it creates parallel compile jobs for each partition.

EXAMPLES

The following example assumes the design *RISC_CORE* is already in memory, and that the current design is *RISC_CORE*. It creates partitions from all designs automatically and uses **-force** to enable the partition since there are multiple instantiated designs.

```
prompt> set_compile_partitions -auto -force
```

In this example, the **report_partitions** command is used to see the results:

```
prompt> report_partitions
```

```
*****
Report   : partitions
Design   : RISC_CORE (unmapped)
Date     : Thu May 10 15:37:37 2001
*****
```

Partition attributes :

```
(*) - partition.
%   - the percentage w.r.t total design area.
d   - dont_touch.
m(n) - multiple instantiated design (number of instantiation).
```

Designs	Attributes
RISC_CORE(*)	8.7%
STACK_TOP	3.9%
STACK_MEM(*)	7.6%, m(3), I_STACK_TOP/I1_STACK_MEM
STACK_FSM	3.2%
REG_FILE(*)	20.7%
DATA_PATH(*)	8.8%
PRGRM_CNT_TOP(*)	6.6%
PRGRM_CNT	2.2%
PRGRM_DECODE	2.2%

PRGRM_FSM	0.9%
CONTROL(*)	5.3%
INSTRN_LAT(*)	8.7%
ALU(*)	18.4%

The following example creates partitions on designs that are in hierarchical level one (subdesigns of the top-level design) and uses **-force** to enable the partition since there are multiple instantiated designs:

```
prompt> set_compile_partitions -level 1 -force
```

SEE ALSO

acs_compile_design(2)
acs_recompile_design(2)
acs_refine_design(2)
report_partitions(2)

set_compile_spg_mode

Sets tool settings to improve timing correlation with either IC Compiler II or IC Compiler. The **set_compile_spg_mode** command settings take effect when you run the **compile_ultra**, **compile_ultra -incremental**, or the **optimize_netlist -area** command.

SYNTAX

```
status set_compile_spg_mode  
[-reset]
```

ARGUMENTS

-reset

Resets the updated settings back to their default values, except user-set values.

DESCRIPTION

The **set_compile_spg_mode** command sets tool settings to improve timing correlation with IC Compiler II or IC Compiler. Specify **icc2** mode to align the settings to the timer settings in IC Compiler II. Specify **icc** mode to align the settings to the timer settings in IC Compiler. For example, in **icc2** mode, the command enables enhanced timing modeling for RC estimation, removes long net pessimism, enables placement pre-clustering and also enables recovery removal checks in Design Compiler Graphical.

The command **set_compile_spg_mode** prints a table of the updated tool settings. You can also get a report of the updated settings by using the **report_compile_spg_mode** command.

The command **set_compile_spg_mode** does not override the tool settings that you have already set.

The Design Compiler Graphical QoR with the **set_compile_spg_mode** command can look pessimistic and needs to be measured post **place_opt** command.

The command updates the settings before optimization starts during **compile_ultra**, **compile_ultra -incremental**, and **optimize_netlist**. The command is supported only in topographical mode.

For detailed information about the values that are changed when you use the **set_compile_spg_mode** command, see the "Using Topographical Technology" chapter in the *Design Compiler User Guide*.

EXAMPLES

The following examples show the usage of the command in different modes:

```
prompt> set_compile_spg_mode icc2  
prompt> compile_ultra
```

```
prompt> set_compile_spg_mode icc  
prompt> compile_ultra
```

```
prompt> set_compile_spg_mode -reset  
prompt> compile_ultra
```

SEE ALSO

- `compile_ultra(2)`
- `optimize_netlist(2)`
- `report_compile_spg_mode(2)`

set_congestion_optimization

Sets the **congestion_optimization** attribute on the specified hierarchical cells or designs, allowing congestion optimization to be performed on the objects. This command is supported only in topographical mode.

SYNTAX

```
status set_congestion_optimization  
  obj_list  
  [true | false]
```

Data Types

obj_list list

ARGUMENTS

obj_list

Specifies a list of hierarchical cell or design names for which to enable congestion optimization. Cell names in *obj_list* must be from the current design. If multiple objects are specified, they must be enclosed in quotation marks (") or braces ({}).

true | false

Specifies the value for the **congestion_optimization** attribute. When the attribute is set to true (the default), congestion optimization is enabled on the specified objects (hierarchical cells or designs). Congestion optimization is performed on the specified objects only if you run the **compile_ultra** command with the **-f-congestion** option.

DESCRIPTION

The **set_congestion_optimization** command sets the **congestion_optimization** attribute on the specified *obj_list*. This attribute is used to specify the hierarchical cells or designs to be optimized for congestion.

Note that the optimizations for congestion are only performed by the **compile_ultra** command with the **-congestion** option.

If a cell with a specified name is found in the current design, the **congestion_optimization** attribute is set to the specified value in the cell. If no cell with a specified name is found, the tool searches for a design and the attribute is set for the design.

If a hierarchical cell is set with the **congestion_optimization** attribute and the cell contains other hierarchical cells, the attribute is propagated recursively on the cells unless they already have the attribute set on them.

EXAMPLES

The following example shows how to enable congestion optimization on hierarchical cells named c0 and c1/m2 and how to disable congestion optimization for the hierarchical cell named c0/m1:

```
prompt> set_congestion_optimization {c0 c1/m2} TRUE  
prompt> set_congestion_optimization c0/m1 FALSE
```

SEE ALSO

- compile(2)
- set_attribute(2)
- get_attribute(2)
- remove_attribute(2)

set_congestion_options

Sets options for congestion optimization.

SYNTAX

```
status set_congestion_options  
  [-max_util value]  
  [-layer name]  
  [-availability value]  
  [-coordinate {X1 Y1 X2 Y2}]
```

Data Types

<i>value</i>	float
<i>name</i>	string
<i>X1</i>	float
<i>Y1</i>	float
<i>X2</i>	float
<i>Y2</i>	float

ARGUMENTS

-max_util *value*

Specifies the maximum utilization factor.

-layer *name*

Specifies the layer name whose availability is reduced.

-availability *value*

Specifies the availability of the routing resource for the layer.

-coordinate {*X1 Y1 X2 Y2*}

Specifies the lower-left and upper-right coordinates for which the congestion options will apply. The numbers are in microns.

DESCRIPTION

Congestion optimization and reporting is available with Design Compiler Graphical. The **set_congestion_options** command sets congestion options for the current design. Congestion occurs because the number of wires going through a region exceeds the

capacity of that region. If the ratio of usage-to-capacity is larger than 1, the region is congested.

The **-max_util** option specifies how densely the tool can pack cells in uncongested regions to remove congestion in congested regions. Set this variable based on how much placeable area is needed. The default maximum utilization is 0.95.

The **-layer** and **-availability** options specify how much of the routing resource for the given layer is available to be used. For example, in a design whose M5 and M6 will be 70% occupied by future P/G routing, you can set the availability of M5 and M6 to be 0.30. This means that even if currently there is no route in M5 and M6, they are considered to be 70% full. If you do not want to use M5 and M6 for signal routing at all, you can use **set_ignored_layer** to mark them as ignored. However, you should still specify the availability in this case, because the tool still needs to know the existence of these future nets to perform a good estimation on the coupling effects of them.

Congestion options can be design wide or regional. If the **-coordinate** option is not specified, the values are design wide. If the **-coordinate** option is specified, the values are only applied to the bounding box specified by the option. Regional congestion options are assigned an ID, which can be used to report or remove the regional options.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **set_congestion_options** command:

```
prompt> set_congestion_options -layer METAL5 -availability 0.5
```

```
prompt> set_congestion_options -layer METAL5 -availability 0.7 \  
-coordinate {0 0 10 10}
```

```
prompt> set_congestion_options -max_util 0.9
```

SEE ALSO

- current_design(2)
- remove_congestion_options(2)
- report_congestion(2)
- report_congestion_options(2)
- set_ignored_layers(2)

set_connection_class

Sets the connection class value on ports.

SYNTAX

```
status set_connection_class  
    connection_class_value  
    object_list
```

Data Types

<i>connection_class_value</i>	string
<i>object_list</i>	list

ARGUMENTS

connection_class_value

Specifies the desired connection class value of ports contained in *object_list*. The *connection_class_value* is a single string value. This string can contain any number of space-separated connection classes (see the example below).

object_list

Specifies ports whose connection classes are set.

DESCRIPTION

The **set_connection_class** command places the **connection_class** attribute on ports of designs being optimized (such as the current design). Connection classes are placed on ports of designs in a technology library by using Library Compiler. The **connection_class** attribute is used to indicate a connection class requirement for the network that is connected to the specified port.

The "connection class" label is an attribute that is used to describe connection requirements for a given technology. Only those loads and drivers with the same connection class label can be legally connected. The "universal" and "default" labels are reserved labels. The "universal" label indicates that a pin or port can legally connect with any other load or driver. The "default" label is used for those library pins that do not otherwise have a connection class assigned to them through any library attributes. For designs being optimized, the "universal" label is assigned to those ports that do not otherwise have an assigned connection class. To change the default connection class label for those ports, use the **default_port_connection_class** variable.

The connection requirements specified using connection classes are treated as Design Rule Constraints by the tool. Optimization attempts to build designs that meet connection class requirements even at the expense of other constraints on the design (such as area or power).

To view connection class values on ports, use the **report_port** or **get_attribute** command. To see the default connection class value for a library, use the **report_lib** command. To check your design for connection class violations, use the **check_design** command. To reset a connection class value, use the **remove_attribute** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets a connection class "internal" to the port named *xyz* in the current design:

```
prompt> set_connection_class internal xyz
```

The following example sets the connection classes "internal" and "external" on the *out* port in the *test* design:

```
prompt> set_connection_class "internal external" test/out
```

In the following example, the connection class on a port is removed:

```
prompt> remove_attribute [get_ports xyz] connection_class
```

SEE ALSO

- [all_outputs\(2\)](#)
- [remove_attribute\(2\)](#)
- [report_lib\(2\)](#)
- [report_port\(2\)](#)
- [reset_design\(2\)](#)
- [default_port_connection_class\(3\)](#)
- [target_library\(3\)](#)

set_constant_register_removal

Sets an attribute to constant register(s) to selectively preserve or optimize it.

SYNTAX

status **set_constant_register_removal**
objects *true | false*
object_list

ARGUMENTS

DESCRIPTION

The **set_constant_register_removal** command is applied on constant sequential cell(s) or hierarchical cell(s) to optimize or preserve it selectively. It set an attribute **remove_constant_register** on specified register. The attribute can store true or false based upon your input. **set_constant_register_removal** when set to false on objects it sets the attribute **remove_constant_register** to value false and does not remove the constant register. When the command is set to true on objects it removes the constant register. You can use the attribute **remove_constant_register** to query the constant registers which are not removed after compile_ultra. This attribute can be used before compile_ultra to query registers which are to be removed. **set_constant_register_removal** honors size_only and issues warning if any register in this category is been set.

EXAMPLES

The following command optimizes register const_regA:

```
prompt> set_constant_register_removal const_regA true
```

The following command preserves register const_regB:

```
prompt> set_constant_register_removal const_regB false
```

The following command issues warning OPT-1221 as register const_sizeonly_regC is size_only:

```
prompt> set_constant_register_removal const_sizeonly_regC false
```

SEE ALSO

set_context_margin

Specifies the margin by which to tighten or relax constraints.

SYNTAX

```
string set_context_margin  
  [-percent]  
  [-relax]  
  [-min]  
  [-max]  
  value  
  [object_list]
```

Data Types

value float
object_list list

ARGUMENTS

-percent

Indicates that the specified value is a percentage of the delay.

-relax

Relaxes the constraint.

-min

Specifies the margin for minimum constraints.

-max

Specifies the margin for maximum constraints.

value

Determines the margin in absolute value or percentage value.

object_list

Specifies a list of cells or pins.

DESCRIPTION

The **set_context_margin** command specifies a margin to add to or to subtract from input and output delay values when the values are generated by the **write_context** command. The margin can be specified as an absolute value or as a percentage of the constraint.

The constraints are created using the **characterize** or **dc_allocate_budgets** command, and then written out using the **write_context** command.

By default, the input and output delays are adjusted so that they are more constraining. This means that the specified margin is added to the maximum delay values and subtracted from the minimum delay values. If you specify the **-relax** option, the margin is subtracted from the maximum delay values and added to the minimum delay values.

The margin applies to the current design when no object list is specified. When determining the margin for a pin, the value set for the pin is used. If you do not specify a value for the pin, the value set for the parent cell is used. If neither value is set, the value set for the current design is used. If no margin is specified, the default value is 0.0

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command adds a margin of 0.5 to all maximum values of input and output delay constraints and subtracts the margin from all minimum values of input and output delay constraints. The margin applies to all objects in the current design.

```
prompt> set_context_margin 0.5
```

The following command relaxes (or reduces) the maximum value input delays on *I2/IN* by 10 percent:

```
prompt> set_context_margin -relax -max -percent 10.0 I2/IN
```

SEE ALSO

characterize(2)
dc_allocate_budgets(2)
write_script(2)

set_cost_priority

Sets the **cost_priority** attribute to a specified value on the current design.

SYNTAX

```
status set_cost_priority  
  [-default]  
  [-delay]  
  cost_list  
  [-design_rules]  
  [-min_delay]
```

Data Types

cost_list list

ARGUMENTS

-default

Removes the **cost_priority** attribute so that the **compile** command uses its default priority.

-delay

Specifies that **max_delay** has higher priority than the maximum design rules.

cost_list

Specifies a list of costs in decreasing order of priority, selected from the following: **max_delay**, **min_delay**, **max_transition**, **max_fanout**, **max_capacitance**, **cell_degradation**, and **max_design_rules**.

-design_rules

Specifies that **max_design_rules** cost has higher priority than the maximum delay cost.

-min_delay

Specifies that **min_delay** has higher priority than the **max_delay**, but lower priority than the maximum design rules.

DESCRIPTION

The **set_cost_priority** command sets the **cost_priority** attribute on the current design. This attribute changes the precedence that

the **compile** command uses when different constraints conflict with each other. For example, by default the design rule **max_transition** has priority over **max_delay**, meaning that if both constraints cannot be met, **compile** fixes the transition violation at the expense of delay.

If a list of costs is given, any costs that are not in the list are assumed to follow afterwards in their default relative priority.

If **set_cost_priority** is specified more than once on a design, the most recent setting is used and the earlier values are removed.

To undo **set_cost_priority**, use the **-default** option, the **remove_attribute** command, or the **reset_design** command.

Use the **report_constraints** command to obtain a summary of the constraints of the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the priority so that design rule violations are not fixed if they increase max delay cost:

```
prompt> set_cost_priority -delay
```

The following example sets the priority so that **max_fanout** and **max_capacitance** have higher priority than delay, but **max_transition** and **cell_degradation** have lower priority:

```
prompt> set_cost_priority {max_fanout max_capacitance max_delay}
```

SEE ALSO

- compile(2)
- current_design(2)
- remove_attribute(2)
- report_compile_options(2)
- report_constraint(2)
- reset_design(2)

set_critical_range

Sets the **critical_range** attribute to a specified value on a list of designs.

SYNTAX

```
status set_critical_range  
    range_value  
    designs
```

Data Types

```
range_value    float  
designs         list
```

ARGUMENTS

range_value

Specifies the value to which the **critical_range** attribute is to be set.

designs

Indicates the list of designs to which the **critical_range** attribute applies.

DESCRIPTION

This command sets the **critical_range** attribute to the specified value on the specified designs.

The tool uses the **critical_range** attribute of the top-level design as the default critical range for path groups that do not have a critical range set. If you do not set the **critical_range** attribute, such path groups get a critical range of 0.0. You can assign critical range values to individual path groups with the **group_path -critical_range** command.

Critical range specifies a margin of delay for path groups in optimization. This must be a positive float number or 0.0. A critical range of 0.0 means that only the most critical paths (the ones with the worst violation) are optimized. If you specify a nonzero critical range, near-critical paths within that amount of the worst path will also be optimized, if possible.

To undo **set_critical_range**, use the **remove_attribute** or **reset_design** command.

To show the critical range values for each path group, use the **report_path_group** command. To show the critical range cost of the design, use the **report_constraint** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example sets the default critical range for the top design to 10.0:

```
prompt> set_critical_range 10.0 top
```

SEE ALSO

- `compile(2)`
- `group_path(2)`
- `remove_attribute(2)`
- `report_constraint(2)`
- `report_path_group(2)`
- `reset_design(2)`

set_current_command_mode

SYNTAX

string **set_current_command_mode**
-mode *command_mode* | -command *command*

Data Types

command_mode string
command string

ARGUMENTS

-mode *command_mode*

Specifies the name of the new command mode to be made current. **-mode** and **-command** are mutually exclusive; you must specify one, but not both.

-command *command*

Specifies the name of the command whose associated command mode is to be made current. **-mode** and **-command** are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **set_current_command_mode** sets the current command mode. If there is a current mode in effect, the current mode is first canceled, causing the associated clean up to be executed. The initialization for the new command mode is then executed as it is made current.

If the name of the given command mode is empty, the current command mode is cancelled and no new command mode is made current.

A current command mode stays in effect until it is displaced by a new command mode or until it is cleared by an empty command mode.

If the command completes successfully, the new current command mode name is returned. On failure, the command returns the previous command mode name which will remain current and prints an error message unless error messages are suppressed.

EXAMPLES

The following example sets the current command mode to a mode called "mode_1"

```
prompt> set_current_command_mode -mode mode_1
```

The following example clears the current command mode without setting a new mode.

```
prompt> set_current_command_mode -mode ""
```

The following example sets the current command mode to the mode associated with the command "modal_command"

```
prompt> set_current_command_mode -command modal_command
```

set_data_check

Sets data-to-data checks using the specified values of setup and hold time.

SYNTAX

```
status set_data_check  
-from from_object  
| -rise_from from_object  
| -fall_from from_object  
-to to_object  
| -rise_to to_object  
| -fall_to to_object  
[-setup | -hold]  
[-clock clock_object]  
[check_value]
```

Data Types

<i>from_object</i>	collection of 1 object
<i>to_object</i>	collection of 1 object
<i>clock_object</i>	collection of 1 object
<i>check_value</i>	float

ARGUMENTS

-from *from_object*

Specifies a pin or port in the current design as the related pin of the data-to-data check to be set. Both rising and falling delays are checked.

You must specify one of **-from**, **-rise_from**, or **-fall_from**.

-rise_from *from_object*

Specifies a pin or port in the current design as the related pin of the data-to-data check to be set. This option applies only to rising delays at the related pin.

-fall_from *from_object*

Specifies a pin or port in the current design as the related pin of the data-to-data check to be set. This option applies only to falling delays at the related pin.

-to *to_object*

Specifies a pin or port in the current design as the constrained pin of the data-to-data check to be set. Both rising and falling delays are constrained.

You must specify one of **-to**, **-rise_to**, or **-fall_to**.

-rise_to to_object

Specifies a pin or port in the current design as the constrained pin of the data-to-data check to be set. This option applies only to rising delays at the constrained pin.

-fall_to to_object

Specifies a pin or port in the current design as the constrained pin of the data-to-data check to be set. This option applies only to falling delays at the constrained pin.

-setup

Indicates that the data check value is for setup analysis only.

-hold

Indicates that the data check value is for hold analysis only.

If neither **-setup** nor **-hold** is specified, the value applies to both setup and hold.

-clock clock_object

Specifies a single clock that launches the signal for the related pin of the data check. The "from" object must be clocked by the specified clock.

check_value

Specifies the value of the setup or hold time for the check.

DESCRIPTION

The **set_data_check** command specifies a data-to-data check to be performed between the "from" (related) object and the "to" (constrained) object using the specified setup or hold value.

The pulse relation between clocks of the related pin and the constrained pin is considered to be zero cycles. The normal sequential check uses one cycle. Data checks on clock pins are not supported.

The data check is treated as nonsequential; that is, the path goes through the related and constrained pins and is not broken at these pins. To report the constraint related to the data check, use the **report_timing** command with the **-to** option to specify the data-check-constrained pin.

To remove information set by the **set_data_check** command, use the **remove_data_check** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example creates a data-to-data check from pin and1/B to pin and1/A with respect to the rising edge of the signal at and1/B, using a setup and hold time of 0.4.

```
prompt> set_data_check -rise_from and1/B -to and1/A 0.4
```

The following example creates a data-to-data check from pin and1/B to pin and1/A with respect to the rising edge of the signal at and1/B, coming from the clock domain of CK1, and constraining only the falling edge of the signal at and1/A, using a setup time of 0.5 and no hold check.

```
prompt> set_data_check -rise_from and1/B -fall_to and1/A \  
-setup -clock [get_clock CK1] 0.5
```

SEE ALSO

- `remove_data_check(2)`
- `report_timing(2)`
- `update_timing(2)`
- `timing_enable_multiple_clocks_per_reg(3)`

set_datapath_gating_options

Specifies whether an instance should be included or excluded for datapath gating processing.

SYNTAX

```
status set_datapath_gating_options
  [-include instance_or_design]
  [-exclude instance_or_design]
  [-reset]
  [-dp]
  [-dw]
  [-nongroup]
  [-retime instance_or_design]
  [-retime_clk_period period_value]
```

Data Types

```
instance_or_design  list
period_value       float
```

ARGUMENTS

-include *instance_or_design*

Specifies a list of instances in the current design to be included for datapath gating. Datapath gating is performed only on the datapath elements in these instances.

-exclude *instance_or_design*

Specifies a list of instances in the current design to be excluded from datapath gating.

-reset

Specifies that the entire current design is to be considered for datapath gating and disregard all previous **-include** and **-exclude** option settings. The **-reset** option is mutually exclusive with the **-include** and **-exclude** options.

-dp

Specifies that only datapath blocks be considered for datapath gating. All the DesignWare singleton instances, even if they are specified in the **-include** list, are disregarded.

-dw

Specifies that only DesignWare singleton instances be considered for datapath gating. All the extracted datapath blocks, even if they are specified in the **-include** list, are disregarded.

-nongroup

Specifies that the gated instances are not ungrouped in the compile flow.

-retime *instance_or_design*

Specifies a list of instances or designs to be gated after retiming. Datapath gating is not performed on these objects before retiming is done.

-retime_clk_period *period_value*

Specifies the clock period used by retiming on the objects listed by **-retime**.

DESCRIPTION

The **set_datapath_gating_options** command sets an attribute on the list of specified instances or designs. The attribute controls whether or not an instance or a design is processed during the datapath gating optimization phase.

Datapath gating is enabled for the entire design by default when you set the **power_enable_datapath_gating** variable to **true**. However, if you explicitly include specific instances or designs for datapath gating using the **-include** option, only the specified instances or designs are considered for datapath gating. The current design and all instances under the current design are ignored.

If the **-dp** option is specified, only extracted datapath designs are considered for gating. All DesignWare singletons are ignored even if they are in the **-include** list. Datapath designs containing operators that appear in the **-exclude** list are ignored.

If the **-dw** option is specified, only DesignWare singleton instances are considered for gating. All extracted datapath cells are ignored. DesignWare singleton instances that appear in the **-exclude** list are ignored. Any DesignWare instance that has been included for gating is extracted to a datapath block.

If the **-retime** option is used, the instances or designs specified are not gated before retiming is done. They are gated after retiming.

If the **-retime_clk_period** option is specified, the clock period is used to guide datapath gating with retiming. This is recommended for a two-pass retiming flow and must be specified in the initial compile prior to retiming.

EXAMPLES

In the following example, datapath gating is performed on the datapath elements in the U4 and U5 instances:

```
prompt> set_datapath_gating_options -include [get_cells {U4 U5}]
```

The following example specifies that the U3/U5 instance should be excluded from the datapath gating processing:

```
prompt> set_datapath_gating_options -exclude [get_cells U3/U5]
```

The following example shows the cumulative behavior of the command:

```
prompt> set_datapath_gating_options -include {a b}
1
```

```
prompt> set_datapath_gating_options -include {c d}
1
```

```
prompt> set_datapath_gating_options -exclude {a}
```

```

1

prompt> set_datapath_gating_options
Included objects { b c d }
Excluded objects { a }
1

```

The following example lists all instances included and excluded for datapath gating:

```

prompt> set_datapath_gating_options
Included objects { U4 U5 }
Excluded objects { U3/U5 }
1

```

The following example removes all previous datapath gating settings:

```

prompt> set_datapath_gating_options -reset
1

prompt> set_datapath_gating_options
No objects included or excluded for gating.
1

```

The following examples show the behavior of the **-dw** and **-dp** options. U1 and U2 are not auto-ungrouped.

In the first example, only datapath blocks inside U2 are tried for gating because DesignWare cells are excluded.

In the second example, only datapath blocks inside U1 are tried for gating and DesignWare cells are excluded from gating.

```

prompt> set_datapath_gating_options -exclude U2 -dw
1

prompt> set_datapath_gating_options -include U1 -dw
1

```

The following example shows how to use the **-retime** option in the two-pass retiming flow. SUB_DESIGN will be retimed in the second pass. The DesignWare blocks in SUB_DESIGN will be gated after the retiming is done.

```

#Turn on datapath gating
set power_enable_datapath_gating true

#specify the sub-designs to be retimed and its retiming clock period
set_datapath_gating_options -retime SUB_DESIGN -retime_clk_period 0.5
compile_ultra

#perform retiming on "SUB_DESIGN"
current_design SUB_DESIGN
optimize_registers

```

The following example shows how to use the **-retime** option in the single-pass global retiming flow. SUB_DESIGN is specified by **set_optimize_registers**, so there is no need to use **-retime** to specify it for incremental datapath gating.

```

#Turn on datapath gating
set power_enable_datapath_gating true

#specify the sub-designs to be retimed
set_optimize_registers true SUB_DESIGN

#top-down compile with global retiming and incremental DG

```

compile_ultra

The following example shows how to use the **-retime** option in the single-pass adaptive retiming flow. The top design is to be retimed.

```
#Turn on datapath gating
```

```
set power_enable_datapath_gating true
```

```
#top-down compile with adaptive retiming and incr. DG
```

```
compile_ultra -retime
```

SEE ALSO

```
report_datapath_gating(2)
```

```
power_enable_datapath_gating(3)
```

set_datapath_optimization_effort

Sets the **datapath_optimization_effort** attribute on specified designs, cells, or references, indicating the level of datapath optimization during **compile_ultra** command activity.

SYNTAX

```
status set_datapath_optimization_effort  
  object_list  
  effort_level
```

Data Types

```
object_list  list  
effort_level string
```

ARGUMENTS

object_list

Specifies a list of cells, references, or designs for the attribute to be set.

effort_level

Specifies the effort level with which to set the **datapath_optimization_effort** attribute. Valid values are **high**, **medium**, **low**. The default value is **high**.

DESCRIPTION

This command sets the **datapath_optimization_effort** attribute on the specified objects so that different levels of datapath optimization can be applied when running the **compile_ultra** command. The **compile** command does not ungroup cells or designs with the **datapath_optimization_effort** attribute.

This attribute is removed with the **remove_attribute** or **reset_design** command.

EXAMPLES

In the following example, the **datapath_optimization_effort** attribute is set to **medium** on the *U1* cell:

```
prompt> set_datapath_optimization_effort U1 medium
```

If the **datapath_optimization_effort** attribute is specified on a reference object, cells using that reference are not ungrouped during **compile**. In this example, all instances of *ADDER* in the current design are not ungrouped:

```
prompt> set_datapath_optimization_effort [get_references ADDER] medium
```

This example shows that a design *ND17* with the attribute set to **high** is ungrouped whenever **compile** encounters the design:

```
prompt> set_datapath_optimization_effort [get_designs ND17] high
```

SEE ALSO

- compile(2)
- current_design(2)
- remove_attribute(2)
- reset_design(2)

set_default_drive

Sets the default driving strength for specified objects, to be used by Top-Down Environmental Propagation (TDEP).

SYNTAX

```
status set_default_drive
  [-min]
  [-max]
  [-rise]
  [-fall]
  [-none]
  [resistance]
  [cell_or_pin_list]
```

Data Types

```
resistance      float
cell_or_pin_list list
```

ARGUMENTS

-min

Indicates that *resistance* is to apply only to minimum resistance.

-max

Indicates that *resistance* is to apply only to maximum resistance. This is the default if no **-min** or **-max** options is specified, and also if both options are given.

-rise

Indicates that *resistance* is to apply only to rise resistance.

-fall

Indicates that *resistance* is to apply only to fall resistance.

-none

Indicates that previous information set by **set_default_drive** is to be removed.

resistance

Specifies the resistance value. Allowed resistance values are any non-negative floating-point numbers.

cell_or_pin_list

Specifies a list of names of cells or pins for which the default driving strength is to be set.

DESCRIPTION

This command specifies a default drive value, to be used later by Top-Down Environmental Propagation (TDEP).

Note that you can specify both maximum and minimum values by issuing two separate commands, one with the **-max** option and one with the **-min** option. However, if you specify both or none of these two options, only the maximum value is set and the minimum value is removed. To always remove both values use the **-none** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets default drives and then removes them by using the **-none** option:

```
prompt> set_default_drive -rise 5 U_L1
Performing set_default_drive on cell 'U_L1'.
1
```

```
prompt> set_default_drive -rise -min 7 U_L1
Performing set_default_drive on cell 'U_L1'.
1
```

```
prompt> set_default_drive -fall 6 U_L1
Performing set_default_drive on cell 'U_L1'.
1
```

```
prompt> set_default_drive -fall -min 8 U_L1
Performing set_default_drive on cell 'U_L1'.
1
```

```
prompt> set_default_drive -none
1
```

SEE ALSO

- derive_constraints(2)
- set_default_driving_cell(2)
- set_default_fanout_load(2)
- set_default_load(2)

set_default_driving_cell

Sets the default driving cell for specified objects, to be used by Top-Down Environmental Propagation (TDEP).

SYNTAX

```
status set_default_driving_cell
  [-lib_cell lib_cell_name]
  [-library lib]
  [-rise]
  [-fall]
  [-pin pin_name]
  [-from_pin from_pin_name]
  [-dont_scale]
  [-no_design_rule]
  [-multiply_by factor]
  [-none]
  cell_or_pin_list
```

Data Types

```
lib_cell_name    string
lib              string
pin_name         string
from_pin_name    string
factor           float
cell_or_pin_list list
```

ARGUMENTS

-lib_cell *lib_cell_name*

Specifies the name of a library cell to be used to drive the ports. Both the **driving_cell_rise** and the **driving_cell_fall** string attributes are set to *lib_cell_name* on the ports. If the cell has more than one output pin, you must also use the **-pin** option. To specify different cells for the rising and falling cases, execute the command twice, once with **-rise** and once with **-fall**, using the appropriate *lib_cell_name* for each.

-library *lib*

Specifies the library in which to find *library_cell_name*. This is either a library name or a collection. You must use this option with the **-lib_cell** option. By default, the libraries in **link_library** are searched for the cell. To specify different libraries for the rising and falling cases, execute the command twice, once with **-rise** and once with **-fall**, using the appropriate *lib* for each.

-rise

Indicates that the *lib_cell_namej*, *lib*, *pin_name*, and *from_pin_name* correspond to the rising case. You can use this option with -

fall to specify both the rising and the falling case.

-fall

Indicates that the *lib_cell_name*, *lib*, *pin_name*, and *from_pin_name* correspond to the falling case. You can use this option with **-rise** to specify both the rising and the falling case.

-pin *pin_name*

Specifies the output pin on the driving cell that is to drive the ports. This pin name is needed if the driving cell has more than one output pin, or if the **-from_pin** option is used. You must use this option with the **-lib_cell** option. The default is to use the first timing arc found on the library cell. To specify different pins for the rising and falling cases, execute the command twice, once with **-rise** and once with **-fall**, using the appropriate *pin_name* for each.

-from_pin *from_pin_name*

Specifies the name of the input pin on the driving cell that is to be used when finding a timing arc. This input pin is needed if the driving cell has more than one input pin and the arcs from those pins have different drive characteristics. You must use this option with both the **-pin** and the **-lib_cell** options. The default is to use the first timing arc found on the library cell.

-dont_scale

Indicates that the timing analyzer is not to scale the drive capability of the ports according to the current operating conditions. By default, the port drive capability is scaled for operating conditions exactly as the driving cell itself would have been scaled. You must use this option with the **-lib_cell** option.

-no_design_rule

Indicates that the design rules associated with the driving cell are not to be applied to the driven port. However, timing-related attributes are still applied. By default, design rule attributes (**max_fanout**, **max_capacitance**, **max_transition**, **min_fanout**, **min_capacitance**, **min_transition**) are derived from the driving cell and its library and applied to the port.

-multiply_by *factor*

Specifies a factor by which to multiply the delay characteristics of the ports. You must use this option with the **-lib_cell** option. The default value is 1.0. Both the load delay and the transition times of the port are affected.

-none

Deletes previous **set_default_driving_cell** information.

cell_or_pin_list

Specifies a list of names of cells or pins to which the information applies.

DESCRIPTION

This command sets a default driving cell for specified cells or pins. This driving cell will later be used by Top-Down Environmental Propagation (TDEP).

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the default driving cell and then removes it using the **-none** option:

```
prompt> set_default_driving_cell -lib_cell ND2 -library lsi_10k \  
-pin Z -from_pin A -rise -dont_scale -mult 2.3;  
1
```

```
prompt> set_default_driving_cell -lib_cell IVP -library lsi_10k \  
-pin Y -from_pin B -fall;  
1
```

```
prompt> set_default_driving_cell -none;  
1
```

SEE ALSO

- derive_constraints(2)
- set_default_drive(2)
- set_default_fanout_load(2)
- set_default_load(2)

set_default_fanout_load

Sets the default fanout load to be used by Top-Down Environmental Propagation (TDEP).

SYNTAX

```
status set_default_fanout_load  
[-none]  
[fanout_load_value]  
[cell_or_pin_list]
```

Data Types

```
fanout_load_value  float  
cell_or_pin_list  list
```

ARGUMENTS

-none

Indicates that previous information set by **set_default_fanout_load** is to be removed.

fanout_load_value

Specifies the fanout load value. Allowed values are any non-negative floating-point numbers.

cell_or_pin_list

Specifies a list of names of cells or pins for which the default fanout load is to be set.

DESCRIPTION

This command specifies a default fanout load value, to be used later by Top-Down Environmental Propagation (TDEP).

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets a default load and then removes it by using the **-none** option:

```
prompt> set_default_fanout_load 2 U_L1/U10  
Performing set_default_fanout_load on cell 'U_L1/U10'.  
1  
  
prompt> set_default_fanout_load -none  
1
```

SEE ALSO

- derive_constraints(2)
- set_default_drive(2)
- set_default_driving_cell(2)
- set_default_load(2)

set_default_input_delay

Sets the value of the input delay as a percentage of the clock period to be assigned during environment propagation.

SYNTAX

```
status set_default_input_delay  
[-none]  
percent_delay  
[cell_or_pin_list]
```

Data Types

<i>percent_delay</i>	float
<i>cell_or_pin_list</i>	collection

ARGUMENTS

-none

Resets any existing default delays.

percent_delay

Specifies the delay as a percentage of the clock period.

cell_or_pin_list

Specifies the list of cells or pins on which the input delay is to be set.

DESCRIPTION

This command sets the value of the input delay as a percentage of the clock period to be assigned during environment propagation. By default, a value of 30 is globally used. This value can be overridden on a global basis, on a cell-by-cell basis, or on a pin-by-pin basis. If no arguments are specified, it sets the global default to the new value. The environment propagation first looks for any values set on the pin. If no values exist, it looks at the cell of the pin, and if no values exist, the global default is used.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example resets existing default delays:

```
prompt> set_default_input_delay -none
```

SEE ALSO

`set_default_output_delay(2)`

set_default_load

Sets the default load to be used by Top-Down Environmental Propagation (TDEP).

SYNTAX

```
status set_default_load
  [-min]
  [-max]
  [-pin_load]
  [-wire_load]
  [-none]
  [value]
  [cell_or_pin_list]
```

Data Types

```
value          float
cell_or_pin_list list
```

ARGUMENTS

-min

Indicates that *value* is to apply only to minimum capacitance.

-max

Indicates that *value* is to apply only to maximum capacitance. This is the default if no **-min** or **-max** option is specified, and also if both options are given.

-pin_load

Indicates that *value* is to apply only to pin capacitance.

-wire_load

Indicates that *value* is to apply only to wire capacitance.

-none

Indicates that previous information set by **set_default_load** is to be removed.

value

Specifies the capacitance value. Allowed values are any non-negative floating-point numbers.

cell_or_pin_list

Specifies a list of names of cells or pins for which the default load is to be set.

DESCRIPTION

This command specifies a default load value, to be used later by Top-Down Environmental Propagation (TDEP).

Note that you can specify both maximum and minimum values by issuing two separate commands, one with the **-max** and one with the **-min** option. However, if you specify both or none of these two options, only the maximum value is set and the minimum value is removed. To always remove both values, use the **-none** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets a default load and then removes it by using the **-none** option:

```
prompt> set_default_load -wire -min 4  
1
```

```
prompt> set_default_load -none  
1
```

SEE ALSO

derive_constraints(2)
set_default_drive(2)
set_default_driving_cell(2)
set_default_fanout_load(2)

set_default_output_delay

Sets the output delay as a percentage of the clock period to be assigned during environment propagation.

SYNTAX

```
status set_default_output_delay  
  [-none]  
  percent_delay  
  [cell_or_pin_list]
```

Data Types

<i>percent_delay</i>	float
<i>cell_or_pin_list</i>	collection

ARGUMENTS

-none

Resets any existing default delays.

percent_delay

Specifies the delay as the percentage of the clock period.

cell_or_pin_list

Specifies the list of cells or pins on which the output delay is to be set.

DESCRIPTION

This command sets the value of the output delay as a percentage of the clock period to be used during environment propagation. By default, a value of 30 is used globally. This value can be overridden on a global basis, on a cell-by-cell basis, or on a pin-by-pin basis. If no arguments are specified, the command sets the global default to the new value. The environment propagation first looks for any values set on the pin. If no values exist, it looks at the cell of the pin, and if no values exist, the global default is used.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example resets any existing default delays:

```
prompt> set_default_output_delay -none
```

SEE ALSO

`set_default_input_delay(2)`

set_delay_estimation_options

Sets the parameters that influence preroute delay estimation. This command is supported only in topographical mode.

SYNTAX

```
status set_delay_estimation_options
  [-min_unit_horizontal_capacitance unit_capacitance]
  [-min_unit_vertical_capacitance unit_capacitance]
  [-min_unit_horizontal_resistance unit_resistance]
  [-min_unit_vertical_resistance unit_resistance]
  [-max_unit_horizontal_capacitance unit_capacitance]
  [-max_unit_vertical_capacitance unit_capacitance]
  [-max_unit_horizontal_resistance unit_resistance]
  [-max_unit_vertical_resistance unit_resistance]
  [-min_unit_horizontal_capacitance_scaling_factor scaling_factor]
  [-min_unit_vertical_capacitance_scaling_factor scaling_factor]
  [-min_unit_horizontal_resistance_scaling_factor scaling_factor]
  [-min_unit_vertical_resistance_scaling_factor scaling_factor]
  [-max_unit_horizontal_capacitance_scaling_factor scaling_factor]
  [-max_unit_vertical_capacitance_scaling_factor scaling_factor]
  [-max_unit_horizontal_resistance_scaling_factor scaling_factor]
  [-max_unit_vertical_resistance_scaling_factor scaling_factor]
  [-min_via_resistance via_resistance]
  [-max_via_resistance via_resistance]
  [-min_via_resistance_scaling_factor scaling_factor]
  [-max_via_resistance_scaling_factor scaling_factor]
  [-density_outside_block density]
  [-default]
```

Data Types

```
unit_capacitance  float
unit_resistance  float
scaling_factor    float
via_resistance    float
density           float
```

ARGUMENTS

-min_unit_horizontal_capacitance *unit_capacitance*

Specifies the minimum horizontal capacitance per unit length for delay calculation.

-min_unit_vertical_capacitance *unit_capacitance*

Specifies the minimum vertical capacitance per unit length for delay calculation.

-min_unit_horizontal_resistance *unit_resistance*

Specifies the minimum horizontal resistance per unit length for delay calculation.

-min_unit_vertical_resistance *unit_resistance*

Specifies the minimum vertical resistance per unit length for delay calculation.

-max_unit_horizontal_capacitance *unit_capacitance*

Specifies the maximum horizontal capacitance per unit length for delay calculation.

-max_unit_vertical_capacitance *unit_capacitance*

Specifies the maximum vertical capacitance per unit length for delay calculation.

-max_unit_horizontal_resistance *unit_resistance*

Specifies the maximum horizontal resistance per unit length for delay calculation.

-max_unit_vertical_resistance *unit_resistance*

Specifies the maximum vertical resistance per unit length for delay calculation.

-min_unit_horizontal_capacitance_scaling_factor *scaling_factor*

Specifies the factor that will be multiplied by the minimum horizontal capacitance per unit length before use in delay calculation.

-min_unit_vertical_capacitance_scaling_factor *scaling_factor*

Specifies the factor that will be multiplied by the minimum vertical capacitance per unit length before use in delay calculation.

-min_unit_horizontal_resistance_scaling_factor *scaling_factor*

Specifies the factor that will be multiplied by the minimum horizontal resistance per unit length before use in delay calculation.

-min_unit_vertical_resistance_scaling_factor *scaling_factor*

Specifies the factor that will be multiplied by the minimum vertical resistance per unit length before use in delay calculation.

-max_unit_horizontal_capacitance_scaling_factor *scaling_factor*

Specifies the factor that will be multiplied by the maximum horizontal capacitance per unit length before use in delay calculation.

-max_unit_vertical_capacitance_scaling_factor *scaling_factor*

Specifies the factor that will be multiplied by the maximum vertical capacitance per unit length before use in delay calculation.

-max_unit_horizontal_resistance_scaling_factor *scaling_factor*

Specifies the factor that will be multiplied by the maximum horizontal resistance per unit length before use in delay calculation.

-max_unit_vertical_resistance_scaling_factor *scaling_factor*

Specifies the factor that will be multiplied by the maximum vertical resistance per unit length before use in delay calculation.

-min_via_resistance *via_resistance*

Specifies the minimum via resistance for delay calculation.

-max_via_resistance *via_resistance*

Specifies the maximum via resistance for delay calculation.

-min_via_resistance_scaling_factor *scaling_factor*

Specifies the factor that will be multiplied by the minimum via resistance before use in delay calculation.

-max_via_resistance_scaling_factor *scaling_factor*

Specifies the factor that will be multiplied by the maximum via resistance before use in delay calculation.

-density_outside_block *density*

Specifies the density based thickness variations at the edge of a block. It defines the density outside a block. The density affects the silicon thickness value. This is called thickness variation (due to density). The thickness change is a function of density and variation tables (defined in TLUPlus). It applies to all layers on which IC Compiler performs density calculation. It applies to all scenarios in multicorner-multimode flow. IC Compiler issues an error if the value specified is less than 0.0 or greater than 1.0. When the value is specified as 0.0 or 1.0, the error is not issued. The default value is 0.0.

-default

Resets all per-unit resistance and capacitance scaling factors and all resistance and capacitance scaling factors to default values.

DESCRIPTION

The **set_delay_estimation_options** command specifies the parameters that influence preroute RC and delay estimation. Typical values for the scaling factors are between 0.5 and 5.0. The unit distance is 1 micron.

Specifying the **-default** option restores all per-unit resistances and capacitances to the library-derived values and all resistance and capacitance scaling factors to 1.0.

The design must be read before applying the delay estimation options using this command.

The scaling factors are scenario-aware. When you create a scenario or set a current scenario, you can use this command to create scaling factors for the current scenario.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example runs the **set_delay_estimation_options** command:

```
prompt> set_delay_estimation_options \
-max_unit_vertical_resistance 0.00004 \
-min_unit_horizontal_capacitance_scaling_factor 2.0 \
-min_via_resistance 0.00001 \
-min_via_resistance_scaling_factor 1.5 \
-max_via_resistance 0.00002 \
-max_via_resistance_scaling_factor 1.5
```

SEE ALSO

`report_delay_estimation_options(2)`

set_design_attributes

Sets the specified UPF attribute and value on the listed cells.

SYNTAX

```
string set_design_attributes  
  [-elements list]  
  [-models list]  
  -attribute name_value_pair
```

Data Types

```
list          list  
name_value_pair  string
```

ARGUMENTS

-elements *list*

Specifies the collection of cells on which the attribute is set. Wildcards are accepted in cell names.

-models *list*

Specifies the collection of models on which the attribute is set. This option is only supported for certain attributes.

-attribute *name_value_pair*

Specifies the name and value of the attribute to be set. This option must be specified exactly one time per command. To specify multiple attributes on the same cell, multiple commands must be used.

This option can be specified multiple times.

DESCRIPTION

This command sets a UPF attribute and its value on a list of cells.

The supported attribute names and their associated values are as follows:

- **correlated_supply_group**
This attribute can be set at the top scope of the design or on any cell instance of the design. The value must be one or more groups of supply net names bracketed by curly braces {}. The value can also be the wildcard character (*) alone, which means all supply nets in the design.

This attribute specifies supply nets whose port state triplets and power state triplets should be considered correlated voltages. The tool considers only the minimum with minimum voltages, only nominal with nominal voltages, and only maximum with maximum voltages, without mixing between minimum, nominal, and maximum. This attribute accepts dot (.) or any cell instance as the value for the **-elements** option.

- **derived_external**

This attribute can be set on supply sets in the design. The value must be Boolean (true or false).

This attribute specifies the supply set to be used as a reference-only supply set. The supply set should not already be used for a power connection. This attribute is used with ports with the **iso_source** and **iso_sink** attributes.

- **derived_iso_strategy**

This attribute can be set on cell instances of the design. The value must be the name of an isolation strategy.

This attribute specifies the isolation strategy name to ensure a unique name for the derived strategies in the power domain. It is used in a hierarchical flow to support location fanout.

- **enable_state_propagation_in_add_power_state**

This attribute can be set at the top scope of the design and also on any hierarchical cell. The value must be **true** or **false**. The default of this attribute is **false**.

When this attribute is set to **true**, the tool propagates the name of the supply set state (specified in the **add_power_state** command) to the functional net. After the state name is propagated to the functional net, you can use the net and the supply set state in the **create_pst** and **add_pst_state** commands. When the value is **true**, usage of the "&&" expression is not allowed in the **supply_expr** option of the **add_power_state** command.

When this attribute is set to **false**, the supply set state name is not propagated to the functional nets. The state name specified in the **add_power_state** command is a property of the supply set, so the name is not propagated to the functional nets. To create power state tables, you must create GROUPS using the **create_power_state_group** command and refer to the supply set state names in the GROUP. You can no longer refer to the state names in the **add_pst_state** command. When the value is **false**, usage of the "&&" expression is allowed.

The allowed attribute configurations are:

1. Attribute is set to **false** for the entire design
2. Attribute is set to **true** for the entire design
3. Attribute is set to **true** for the block, and set to **false** for the top design

The following configuration is NOT allowed: Attribute is set to **false** for the block, and set to **true** for the top design.

- **external_supply_map**

This attribute can be set on cell instances of the design and the top design. The value must be the name of a supply set that is already created.

This attribute specifies the mapping between the reference-only supply set and the actual supply set. When this attribute is specified on the design, its value will be the reference-only supply set in the design. When this attribute is specified on the cell instance of the design, its value will be the actual supply set for which the reference-only supply set was created inside the design. This attribute is used with ports with the **iso_source** and **iso_sink** attributes.

- **lower_domain_boundary**

This attribute can be set at the top scope of the design and also on any hierarchical cell. The value must be **true** or **false**.

When this attribute is set to **true**, the tool considers the domain boundary between a domain and another domain that is contained in a lower-level block as the scope of the first (higher hierarchical level) domain.

- **UPF_is_hard_macro**

This attribute can be set on library cells having the **is_macro_cell** Liberty attribute. The value must be **true** or **false**.

If a library cell has the **is_macro_cell** Liberty attribute, then it is considered as a hard macro. When this design attribute is set to

false on such a library cell, the tool does not consider the library cell as a hard macro.

This attribute must be specified together with **-models** option.

- **merge_domain**

This attribute can be set on cell instances of the design. The value must be **true** or **false**.

If set to **true**, this attribute implies that the elements that belong to the same power domain can be merged. The cells specified in the **-elements** option should not be the root cell of a power domain.

- **SNPS_default_power_upf2sv_vct** and **SNPS_default_ground_upf2sv_vct**

These attributes specify the default for the UPF-to-SystemVerilog value conversion table. These attributes can be used with the **-models** option. The value must be set to one of the following: SV_TIED_HI, SV_TIED_LO, UPF_GNDZERO2SV_LOGIC, UPF2SV_LOGIC

- **SNPS_default_power_upf2vh_vct** and **SNPS_default_ground_upf2vh_vct**

These attributes specify the default for the UPF-to-VHDL value conversion table. These attributes can be used with the **-models** option. The value must be set to one of the following: VHDL_TIED_HI, VHDL_TIED_LO, UPF2VHDL_SL, UPF_GNDZERO2VHDL_SL

- **SNPS_default_power_sv2upf_vct** and **SNPS_default_ground_sv2upf_vct**

These attributes specify the default for the SystemVerilog-to-UPF value conversion table. These attributes can be used with the **-models** option. The value must be set to one of the following: SV_LOGIC2UPF, SV_LOGIC2UPF_GNDZERO

- **suppress_iss**

This attribute can be set at the top scope of the design. The value must be the name of one or more power domains. The value can also be the wildcard character (*) which means all power domains in the design. It specifies the domain names where supply set handles are not to be created. This attribute does not accept the **-elements** option.

- **terminal_boundary**

This attribute can be set at the top scope of the design and also on any cell in the list of **-elements** of the command **create_power_domain**. Typically, it is set on a block design, black box, or ETM cell. The value must be **true** or **false**.

When this attribute is set to **true**, the tool considers the boundary of the cell as a terminal and checks, in the **check_mv_design** command, that the driver/load pin's related supply is consistent with the driver/receiver supply defined on the boundary pin (either by **set_port_attributes -driver_supply -receiver_supply** or by **set_related_supply_net**).

- **upf_chip_design** (used on the top-level design)

This attribute can be set on the top-level design. The value must be **true** or **false**. It is used in the isolation strategy definition to support location fanout.

- **relax_constant_corruption_for_macro_inputs_and_primary_outputs** (used on the top-level design)

This attribute can only be set on the top-level design. The value must be **true** or **false**. With default value **false**. If 'TRUE' is set, it is used to skip isolation insertion in literal constants directly driving macro inputs or primary output ports if the clamp value of the strategy matches the logic value of the constant.

- **hetero_fanout_isolation**

The value must be **true** or **false**. When this attribute is **true**, the isolation strategy of location self or parent can insert the isolation cells at the sub-branches, when those branches have the proper sink supply as the isolation strategy specifies.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the **external_supply_map** attribute on the root cell of power domain PD2.

```
prompt> set_design_attributes \
  -elements [get_cells -of_objects [get_power_domains PD2]] \
  -attribute external_supply_map {SS2}
1
```

The following example sets the **derived_external** attribute on supply set SS1.

```
prompt> set_design_attributes \
  -elements {SS1} \
  -attribute derived_external true
1
```

The following example sets the **derived_iso_strategy** attribute on mid11 with power domain MID11.

```
prompt> set_design_attributes \
  -elements {mid11} \
  -attribute derived_iso_strategy iso1
1
prompt> set_isolation iso1 \
  -domain MID11 -source sset1 \
  -elements {mid11/*} \
  -isolation_supply_set sset3
1
prompt> set_isolation_control iso1 \
  -domain MID11 \
  -isolation_signal in1 \
  -location fanout
1
```

The following example sets the **merge_domain** attribute to **true** on cell mid1.

```
prompt> set_design_attributes \
  -elements {mid1} \
  -attribute merge_domain true
1
```

The following example sets the **suppress_iss** attribute to suppress implicit supply set for domain inner/PD_INNER specified for design 'top'.

```
prompt> set_design_attributes \
  -attribute suppress_iss "inner/PD_INNER"
1
```

The following example sets the **upf_chip_design** attribute on a top-level design.

```
prompt> set_design_attributes -elements {.} \
  -attribute upf_chip_design true
1
```

The following example sets the **SNPS_default_power_sv2upf_vct** attribute SV_LOGIC2UPF on model bot and libcell GTECH_NOT.

```
prompt> set_design_attributes \
  -models {bot GTECH_NOT} \
  -attribute SNPS_default_power_sv2upf_vct SV_LOGIC2UPF
1
```

The following example sets the **correlated_supply_group** attribute to specify correlated voltage range comparison for port state

triplets.

If a driver is powered by VDD1 and its load is powered by VDD2, a level shifter is not required because VDD1 and VDD2 are in the same correlated supply group and therefore treated as correlated voltage range during voltage comparison.

However, if the load is powered by VDD3, a level shifter is required because VDD1 and VDD3 are not in the same correlated supply group and therefore treated as independent voltage range during voltage comparison.

```
prompt> add_port_state VDD1 -state {HV 0.8 1.0 1.2}
1
prompt> add_port_state VDD2 -state {HV 0.8 1.0 1.2}
1
prompt> add_port_state VDD3 -state {HV 0.8 1.0 1.2}
1
prompt> add_port_state VDD4 -state {HV 0.8 1.0 1.2}
1
prompt> set_design_attributes \
  -elements {.} \
  -attribute correlated_supply_group "{VDD1 VDD2} {VDD3 VDD4}"
1
```

The following example sets the **lower_domain_boundary** attribute to **true** on the top scope and cell mid1.

```
prompt> set_scope /
1
prompt> set_design_attributes -elements {.} -attribute lower_domain_boundary true
Information: Resolving '.' to the current scope 'top'.
1
prompt> set_design_attributes -elements {mid1} -attribute lower_domain_boundary true
1
```

The following example sets the **enable_state_propagation_in_add_power_state** attribute to **false** on the top scope and to **true** on cell mid1.

```
prompt> set_scope
1
prompt> set_design_attributes \
  -elements {.} \
  -attribute enable_state_propagation_in_add_power_state false
1
prompt> set_design_attributes \
  -elements {mid1} \
  -attribute enable_state_propagation_in_add_power_state true
1
```

The following example sets the **terminal_boundary** attribute to **true** on the top scope and cell mid1.

```
prompt> set_scope
1
prompt> set_design_attributes \
  -elements {.} \
  -attribute terminal_boundary true
1
prompt> set_design_attributes \
  -elements {mid1} \
  -attribute terminal_boundary true
1
```

The following example sets the hard-macro attribute on model "model1" to **TRUE**.

```
prompt> set_design_attributes -models {model1} -is_hard_macro TRUE  
1
```

SEE ALSO

set_port_attributes(2)
set_related_supply_net(2)

set_design_license

Adds license information to the current design and can be used to require a license before a design can be read in.

SYNTAX

```
status set_design_license  
  [-dont_show references]  
  [-quiet]  
  [-limited limited_keys]  
  regular_keys
```

Data Types

```
references    list  
limited_keys  list  
regular_keys list
```

ARGUMENTS

-dont_show *references*

Specifies a list of references that will not be displayed.

-quiet

Specifies that no informational messages will be displayed.

-limited *limited_keys*

Specifies a list of licenses that provide limited access to the design.

regular_keys

Specifies a list of licenses that provide regular access to the design.

DESCRIPTION

This command adds license information to a design. For more information on licensing designs, see the section on licensing in the DesignWare manual.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In this example, the full license *ADDERS* and the limited license *EVAL* is added to the current design. In addition, **-dont_show** indicates that the design will not become visible until all existing references are changed.

```
prompt> set_design_license ADDERS \  
-limited EVAL -dont_show ""
```

SEE ALSO

`list_licenses(2)`

set_design_top

Specifies the top-level design instance.

SYNTAX

```
status set_design_top  
      instance_name
```

Data Types

instance_name string

ARGUMENTS

instance_name

Specifies the top-level instance in the design.

DESCRIPTION

The **set_design_top** command specifies the top-level design instance. This information is used only by simulation and verification tools.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following example shows how to use the **set_design_top** command:

```
prompt> set_design_top ALU07
```

set_dft_clock_controller

Specifies the parameters for DFT-inserted clock controllers.

SYNTAX

```
status set_dft_clock_controller
[-cell_name controller_cell_name]
-design_name controller_design_name
[-ateclocks port_list]
[-pllclocks pin_list]
[-1x_clocks pin_list]
[-2x_clocks pin_list]
[-4x_clocks pin_list]
[-chain_count number_chains]
[-cycles_per_clock number_cycles]
[-test_mode_port port_name]
```

Data Types

<i>controller_cell_name</i>	string
<i>controller_design_name</i>	string
<i>port_list</i>	list
<i>pin_list</i>	list
<i>number_chains</i>	integer
<i>number_cycles</i>	integer
<i>port_name</i>	string

ARGUMENTS

-cell_name *controller_cell_name*

Specifies the instance name of the clock controller that will be inserted by the **insert_dft** command; for example, *pll_controller1*.

-design_name *controller_design_name*

Specifies the design name to insert a clock controller from the DFT library. The only allowed value is **snps_clk_mux**. Custom-designed clock controllers are not supported by this command. Note that the command is accepted without specifying **-design_name**, but it returns incorrect data to the DFT architect.

-ateclocks *port_list*

Specifies one slow clock used for scan shifting that you want to connect to the DFT clock controller. If the clock signal is a reference clock for the on-chip clock generator, the signal must be previously defined by using the **set_dft_signal** with the **-type Oscillator** option. Note that you can only specify one clock because custom-designed clock controllers are not supported by this command.

-pllclocks *pin_list*

For asynchronous OCC controllers, specifies the ordered list of PLL output clock pins to control. The pins must be previously defined as oscillator signals using the **set_dft_signal -type Oscillator** command.

This option is mutually exclusive with the **-1x_clocks**, **-2x_clocks**, and **-4x_clocks** options.

-1x_clocks *pin_list*

For synchronous OCC controllers, specifies the ordered list of PLL output clock pins to control that run at the slowest frequency. When inserting synchronous OCC controllers, this option is required; the first clock in the list is used as the master synchronization clock. The pins must be previously defined as oscillator signals using the **set_dft_signal -type Oscillator** command.

This option is mutually exclusive with the **-pllclocks** option.

-2x_clocks *pin_list*

For synchronous OCC controllers, specifies the ordered list of PLL output clock pins to control that run at two times the slowest frequency. The pins must be previously defined as oscillator signals using the **set_dft_signal -type Oscillator** command.

This option is mutually exclusive with the **-pllclocks** option.

-4x_clocks *pin_list*

For synchronous OCC controllers, specifies the ordered list of PLL output clock pins to control that run at the slowest frequency. The pins must be previously defined as oscillator signals using the **set_dft_signal -type Oscillator** command.

This option is mutually exclusive with the **-pllclocks** option.

-chain_count *number_chains*

Specifies the number of clock chains to insert per clock controller. The default is 1.

-cycles_per_clock *number_cycles*

Specifies the maximum number of capture cycles per clock. If not specified, the default is 1. Capture cycles are cycles where capture clocks are pulsed.

-test_mode_port *port_name*

Specifies the test mode port used to enable the clock controller. Use this option if you have multiple test mode ports and you want to use a specific port to enable the clock controller. The specified port must be defined as a TestMode signal using the **set_dft_signal** command.

DESCRIPTION

The **set_dft_clock_controller** command specifies the clock controller characteristics for the **insert_dft** command to insert and connect clock controllers and clock chains into the design.

A clock controller is a design that controls which clock signals will propagate into scan chains during scan shifting and capture. The clock controller selects between fast clocks coming from an on-chip clock generator (usually a phase-locked loop) and slow clocks coming from external ports.

A clock chain is a scan chain segment of one or more scannable control registers. This chain allows for a per-pattern clock selection mechanism by ATPG. Clock selection values are loaded into the clock chain as part of the regular scan load process.

The on-chip clock generator should already be connected, either directly or indirectly, to the intended sequential elements in the

clock generator clock domain.

For asynchronous OCC controllers, specify the PLL clocks with the **-pllclocks** option. For synchronous OCC controllers, specify the clocks with the **-1x_clocks**, **-2x_clocks**, and **-4x_clocks** options. You cannot mix the asynchronous and synchronous options in the same **set_dft_clock_controller** command, but you can issue multiple **set_dft_clock_controller** commands for separate asynchronous and synchronous OCC controllers.

When clock controller insertion has been configured with the **set_dft_clock_controller** command, the **insert_dft** command performs the following steps:

1. Instantiates a clock controller design and clock chain design.
2. Inserts the clock controller in the clock path between the on-chip clock generator and the sequential cells that it drives.
3. Connects the clock chain to the clock controller.
4. Proceeds with the remainder of DFT insertion.

When a clock controller is inserted in the design, you are responsible for validating that this functionality works in the context of your design.

In high X-tolerance DFTMAX compression flows, the presence of clock chains affects the scan-in and scan-out pin requirements. For more information, see the TEST-1722 man page.

EXAMPLES

The following command set inserts a clock controller and clock chain into the design:

```
prompt> set_dft_configuration -clock_controller enable

prompt> set_dft_signal -view existing_dft -type ScanClock \
    -port slow_clock -timing {45 55}

prompt> set_dft_signal -view existing_dft -type Oscillator \
    -port slow_clock

prompt> set_dft_signal -view existing_dft -type Oscillator \
    -hookup_pin PLL/pll_fast_clock

prompt> set_dft_clock_controller -cell_name snps_pll_controller \
    -design_name snps_clk_mux -cycles_per_clock 2 -chain_count 1 \
    -pllclocks {PLL/pll_fast_clock} -ateclocks {slow_clock}

prompt> insert_dft
```

SEE ALSO

insert_dft(2)
 set_dft_configuration(2)
 set_dft_drc_configuration(2)
 set_dft_signal(2)

write_test_protocol(2)

set_dft_clock_gating_configuration

Specifies the clock-gating configuration for a design.

SYNTAX

```
status set_dft_clock_gating_configuration  
[-exclude_elements object_list]  
[-dont_connect_cgs_of cell_list]
```

Data Types

object_list cells, lib_cells, or clocks
cell_list cells

ARGUMENTS

-exclude_elements *object_list*

Specifies a list of clock-gating cells, clock-gating observation cells, hierarchical cell instances containing these cells, lib_cells of these cells, or clocks of these cells that are to be excluded when the test control pins of clock-gating cells are connected during **insert_dft**. These excluded clock-gating cells are not checked for unconnected test pins during **dft_drc**, and therefore they do not cause TEST-130 violations. Downstream registers driven by these excluded clock-gating cells will have DRC violations if their clocks are uncontrollable.

If a hierarchical cell instance is specified, all clock-gating cells and clock-gating observation cells in that instance are included in the specification.

If a clock-gating library cell is specified, all instances of that cell are included in the specification.

If a clock is specified, clock-gating cells in the respective clock domain are included in the specification. The clock must be defined with the **create_clock** command.

Cells with CTL models or inside CTL models are ignored and excluded from the specification.

This option accepts wild cards and collection inputs. This option is cumulative across multiple commands.

-dont_connect_cgs_of *cell_list*

Specifies a list of registers or hierarchical cell instances containing registers, whose upstream clock-gating cells are to be excluded from test pin connections during **insert_dft**. An upstream clock-gating cell is only excluded from test pin connections if it does not drive any valid scan cells.

If a hierarchical cell instance name is specified, the specification applies to all registers in that instance.

Cells with CTL models or inside CTL models are ignored and excluded from the specification.

If an upstream clock-gating cell of a specified register also drives a CTL modeled cell then its test pin will be connected.

If DFT Compiler cannot find the upstream clock-gating cells for flip-flops during the `preview_dft` or `insert_dft` commands, it issues TEST-154 informational messages for those flip-flops.

This option also accepts wild cards and collection inputs. This option is cumulative across multiple commands.

DESCRIPTION

This command specifies the clock-gating configuration for DFT flows. It affects how clock-gating logic is handled by the **dft_drc** and **insert_dft** commands.

The **-exclude_elements** and **-dont_connect_cgs_of** options both provide methods for excluding the test pins of clock-gating cells from being connected to global test signals during DFT insertion.

Exclusion works by leaving the default deasserted constant value connected to the test pins of excluded clock-gating cells. If an exclusion applies to a clock-gating cell with an existing test pin connection other than the default deasserted constant value, that existing test pin connection remains in place.

The precedence of clock-gating cell test pin connection control methods is as follows, in order of highest to lowest priority:

- **set_dft_connect** (highest priority)
- **set_dft_clock_gating_configuration -exclude_elements**
- **set_dft_clock_gating_configuration -dont_connect_cgs_of** (lowest priority)

When the **-dont_connect_cgs_of** option is used, upstream clock-gating cells of the specified registers are excluded from clock-gating cell test pin connection only if none of the downstream registers are valid scan cells. When the **-exclude_elements** option is used, the test pin connections of the specified clock-gating cells are suppressed even if there are downstream valid scan cells.

The **-exclude_elements** and **-dont_connect_cgs_of** exclusion options will function with, but are not intended for, registers driven by user-defined clock-gating cells defined via **set_dft_clock_gating_pin** command. If you do not want to connect the test pins of these clock-gating cells, then they should not be defined. Registers driven by user-defined clock gating cells are traced through simple buffer and inverter logic only.

EXAMPLES

The following example excludes a particular clock-gating cell from having its test pin connected:

```
prompt> set_dft_clock_gating_configuration \
        -exclude_elements I_ADD/clock_gate_add_out_reg
```

The following example excludes clock-gating cells and clock-gating observation logic inside a hierarchical cell instance:

```
prompt> set_dft_clock_gating_configuration -exclude_elements U_block
```

The following example excludes all instances of a particular library cell of the 'slow' library:

```
prompt> set_dft_clock_gating_configuration \
        -exclude_elements [get_lib_cell slow/ABCD123XYZ]
```

The following example excludes a particular clock-gating observation leaf cell:

```
prompt> set_dft_clock_gating_configuration -exclude_elements Uclk_gate_obs
```

The following example excludes clock-gating cells in a particular clock domain:

```
prompt> create_clock -period 5 CLK
```

```
prompt> set_dft_clock_gating_configuration -exclude_elements CLK
```

The following example excludes any clock-gating cells driving register DATA_IN_reg if the upstream clock-gating cell does not drive any other valid scan cells.

```
prompt> set_dft_clock_gating_configuration -dont_connect_cgs_of DATA_IN_reg
```

SEE ALSO

`reset_dft_clock_gating_configuration(2)`
`report_dft_clock_gating_configuration(2)`

set_dft_clock_gating_pin

Specifies the test pin of a clock-gating cell in a design. The main purpose of this command is to identify the unconnected test pins of the clock-gating cells that were not inserted by Power Compiler. These pins are connected to test ports when you run the **insert_dft** command.

SYNTAX

```
status set_dft_clock_gating_pin  
  object_list  
  -pin_name instance_pin_name  
  [-control_signal ScanEnable | TestMode | scan_enable | test_mode]  
  [-active_state 1 | 0]
```

Data Types

object_list list

ARGUMENTS

object_list

Specifies the clock-gating cell instances for which test pins are specified. This is a required argument.

Wildcards and collections are supported.

-pin_name *instance_pin_name*

Specifies the name of the test pin on the specified instances. This pin name is common to all specified instances. This is a required argument.

-control_signal ScanEnable | TestMode | scan_enable | test_mode

Specifies the type of control signal required by the test pin. The **insert_dft** command connects the pin to a test port of the specified type. This argument is optional. By default, the control signal type is ScanEnable.

-active_state 1 | 0

Specifies the active state of the test pin. This argument is optional. By default, the active state is high, that is, 1.

DESCRIPTION

Specifies the test pins of clock-gating cells not inserted by Power Compiler for DFT flows. The command affects how clock-gating

logic is handled by the **dft_drc** and the **insert_dft** commands.

Identified cells can be connected to user-specified test ports by using the **set_dft_connect** command. However, they are not connected when clock-domain-based connection is done, by using the **set_dft_connect** command. For clock-domain-based connection, only clock-gating cells recognized and inserted by Power Compiler are supported.

You must be careful when specifying the instance pin name because the test pin name is common to all specified instances.

There is no checking for the specified cells. It is your responsibility to make sure that the specified cell and pin are actually a clock-gating cell and test pin that was not identified by Power Compiler. Specifying a cell that is not a clock gating cell can cause undesired results when you run the **dft_drc** and **insert_dft** commands.

Existing connections are maintained. If the specified test pin is already connected to a test port or logic, the **insert_dft** command does not disconnect and reconnect the test pin. Also, the **dft_drc** command will not issue a TEST-130 message for the pin. Only test pins driven by a logic constant and not connected to a net or to a net without a driver are handled by the **dft_drc** command and subsequently connected by the **insert_dft** command.

The **-control_signal** option supports two different keyword conventions. The **ScanEnable** and **TestMode** values are consistent with the **set_dft_signal** command, while the **scan_enable** and **test_mode** values are consistent with the **set_clock_gating_style** command. Either convention can be used.

EXAMPLES

The following examples identify a particular clock-gating cell.

Identification using the default setting:

```
prompt> set_dft_clock_gating_pin {sub1/clk_gate_out1_reg} \
-pin_name TE
```

Identification using explicit specifications:

```
prompt> set_dft_clock_gating_pin {sub1/clk_gate_out1_reg} \
-pin_name TE -control_signal ScanEnable -active_state 1
```

```
prompt> set_dft_clock_gating_pin {sub2/clk_gate_out1_reg} \
-pin_name TE -control_signal ScanEnable -active_state 0
```

Identification using explicit specifications with the test pin controllable by a test mode:

```
prompt> set_dft_clock_gating_pin {sub3/clk_gate_out1_reg} \
-pin_name TE -control_signal TestMode
```

```
prompt> set_dft_clock_gating_pin {sub4/clk_gate_out1_reg} \
-pin_name TE -control_signal TestMode -active_state 0
```

Identify test pin SE on all instances of library cell MyCGCell with default settings

```
prompt> set_dft_clock_gating_pin [get_cells * -hier \
-filter "@ref_name == MyCGCell"] -pin_name SE
```

SEE ALSO

dft_drc(2)
insert_dft(2)
remove_dft_clock_gating_pin(2)
report_dft_clock_gating_pin(2)

set_dft_configuration

Sets the DFT configuration for the current design.

SYNTAX

```
status set_dft_configuration
  [-scan enable | disable]
  [-fix_clock enable | disable]
  [-fix_set enable | disable]
  [-fix_reset enable | disable]
  [-fix_xpropagation enable | disable]
  [-fix_bus enable | disable]
  [-fix_bidirectional enable | disable]
  [-testability enable | disable]
  [-test_points enable | disable]
  [-wrapper enable | disable]
  [-boundary enable | disable]
  [-bsd enable | disable]
  [-clock_controller enable | disable]
  [-mode_decoding_style binary | one_hot]
  [-scan_compression enable | disable]
  [-streaming_compression enable | disable]
  [-pipeline_scan_data enable | disable]
  [-connect_clock_gating enable | disable]
  [-integration enable | disable]
  [-power_control enable | disable]
  [-ieee_1500 enable | disable]
  [-control_points enable | disable]
  [-observe_points enable | disable]
```

ARGUMENTS

-scan enable | disable

Enables or disables scan insertion.

The default is **enable**.

-fix_clock enable | disable

Enables or disables clock AutoFixing. By default, clocks are not AutoFixed.

-fix_set enable | disable

Enables or disables asynchronous set AutoFixing. By default, asynchronous set signals are not AutoFixed.

-fix_reset enable | disable

Enables or disables asynchronous reset AutoFixing. By default, asynchronous resets are not AutoFixed.

-fix_xpropagation enable | disable

Enables or disables X-propagation AutoFixing. By default, X-propagation is not AutoFixed.

-fix_bus enable | disable

Enables or disables three-state bus AutoFixing. By default, three-state buses are AutoFixed.

-fix_bidirectional enable | disable

Enables or disables bidirectional pad AutoFixing. By default, bidirectional pads are AutoFixed.

-testability enable | disable

Enables or disables the testability test-point client, which consists of:

- SpyGlass automatic test-point insertion
 - Configured with the **set_testability_configuration** command
 - Requires a SpyGlass license
- User-defined test-point insertion
 - Configured with the **set_test_point_element** command
 - Requires a DFTMAX license

This option, along with the **set_testability_configuration** command, replaces the deprecated **test_points** client. If the **-testability** option is enabled, those deprecated commands and options are ignored.

-test_points enable | disable

Enables or disables legacy automatic test point insertion. By default, it is disabled. When enabled, to automatically insert test points, you must also configure one or more test point targets with the **set_test_point_configuration** command.

This feature has been deprecated; use the **testability** test-point client instead.

A DFTMAX license is required to enable this feature.

-wrapper enable | disable

Enables or disables the Core wrapper and Shadow LogicDFT utilities. The specifications for the utilities are set with the **set_wrapper_configuration** command.

-boundary enable | disable

Enables or disables the integration of BIST or scan cores using ANSI/IEEE Std 1149.1-1 compliant boundary-scan circuitry. The newly-synthesized circuitry is unmapped, and must be mapped using the **compile** command.

-bsd enable | disable

Enables or disables the insertion of IEEE Std 1149.1/1149.6 compliant boundary-scan circuitry.

-clock_controller enable | disable

Enables or disables the clock controller for on chip clocking.

-mode_decoding_style binary | one_hot

Specifies the type of port decoding to be used for the test modes controller. Allowed values are **binary** and **one_hot**. Specify **binary** to decode ports for the test controller, or **one_hot** to not decode ports for the test controller. If n test modes are specified for the design, specifying **binary** allows mode architecting to use $\text{ceiling}(\log_2(n))$ ports for the test controller. The **one_hot** mode decoding style allows mode architecting to use $n-1$ ports for the test controller.

The default value is **binary**.

-scan_compression enable | disable

Enables or disables the insertion of combinational or serialized scan compression structures.

-streaming_compression enable | disable

Enables or disables the insertion of streaming scan compression structures.

-pipeline_scan_data enable | disable

Enables or disables the insertion of pipeline scan data registers to scan chains. By default, the pipeline scan data utility is disabled.

-connect_clock_gating enable | disable

Enables or disables checking for unconnected test pins of clock-gating cells during **dft_drc** and their connection in **insert_dft**. The default value is **enable**.

-integration enable | disable

Enables or disables the core integration functionality.

-power_control enable | disable

Enables or disables the power controller override functionality.

The power controller override functionality is disabled by default.

When enabled, the **insert_dft** command inserts a wrapper chain override logic along the control signal outputs of the specified power controller block, inside the level of hierarchy that contains the specified power controller block.

This logic is known as the power controller override logic. This logic allows critical power control signals, such as power switch control, isolation control and state retention control signals, to be controlled during test mode without the need for manually-created power controller initialization vectors. It also provides observability of the power controller outputs for improved test coverage.

-ieee_1500 enable | disable

Enables or disables the IEEE 1500 test-mode control functionality.

Use this option to enable both the DFT-inserted IEEE 1500 controller flow and the existing IEEE 1500 controller flow.

DESCRIPTION

The **set_dft_configuration** command specifies the DFT configuration for a design. The DFT configuration is specific to the current design. If you change the DFT configuration and then change the current design, your changes are no longer visible.

Use the **report_dft_configuration** command to display the current DFT configuration of the design.

Use the **reset_dft_configuration** command to reset the current DFT configuration on the design.

Setting the **set_dft_configuration** command with the **-scan disable -clock_gating enable** options and running the **insert_dft** command results in clock-gating cell routing only.

EXAMPLES

The following command enables the clock Autofix utility:

```
prompt> set_dft_configuration -fix_clock enable
```

The following command disables the check for unconnected test pins of clock-gating cells during **dft_drc** and their connection in **insert_dft**:

```
prompt> set_dft_configuration -connect_clock_gating disable
```

SEE ALSO

- insert_dft(2)
- report_dft_configuration(2)
- set_autofix_configuration(2)
- set_autofix_element(2)
- set_dft_signal(2)
- set_testability_configuration(2)
- set_test_point_configuration(2)

set_dft_connect

Specifies a list of DFT connectivity associations. The specified source is connected to objects specified in the target list.

SYNTAX

```
status set_dft_connect  
  string_label  
  -type clock_gating_control | scan_enable  
  -source port_or_pin  
  [-target object_list]  
  [-rise_target rise_clock_list]  
  [-fall_target fall_clock_list]  
  [-exclude cell_list]
```

Data Types

<i>string_label</i>	string
<i>port_or_pin</i>	string
<i>object_list</i>	string
<i>rise_clock_list</i>	string
<i>fall_clock_list</i>	string
<i>cell_list</i>	string

ARGUMENTS

string_label

Specifies an identifier for the connectivity association. Each association must be unique. If the same label is specified with two connectivity associations, the earlier specification is overwritten.

-type clock_gating_control | scan_enable

Specifies the type of connectivity association described by the command.

If **-type** is specified as **scan_enable**, you must also use either the **-target**, **-rise_target**, or **-fall_target** option. If **-type** is specified as **clock_gating_control**, the **-target** option is not required.

-source port_or_pin

Specifies a DFT signal driver port or pin to be connected to the target cells.

For a DFT signal with a hookup pin, its port must be specified as driver. For an internal pin DFT signal, the driver must be the hookup pin.

For **-type clock_gating_control**, the driver must be defined as ScanEnable or TestMode with the **-usage clock_gating** option of the **set_dft_signal** command.

For **-type scan_enable**, the driver must be defined as ScanEnable with the **-usage scan** switch of the **set_dft_signal** command.

-target *object_list*

Specifies a list of objects to be connected to the specified DFT signal driver port or pin. Specified objects can be a list of cells, designs, clocks, or ScanEnable/TestMode pin/port on CTL modeled cells.

For **-type clock_gating_control**, the *cell_list* must be a list of either clock-gating cells, designs containing clock-gating cells, clock-gating observation cells, or a list of clocks *clock_list*.

For **-type scan_enable**, the *cell_list* must be a list of either valid scan cells or designs containing valid scan cells.

A specified pin/port can be only a clock or pin (ScanEnable/TestMode) on a CTL modeled cell.

When a clock or list of clocks *clock_list* is specified, the tool selects appropriate elements in the respective clock domain, based on the connectivity type. Specified clocks must be valid DFT clocks defined with the **set_dft_signal** command. In addition, for **-type clock_gating_control**, the clocks must be defined with the **create_clock** command.

When doing clock-domain-based scan enable connections, the CTL model of the modeled cell/design must contain the relation between a Scan enable and the clock domain. DFT Compiler puts this relation in the model when doing scan insertion at the block level. For hierarchical scan synthesis, this relation is utilized to perform clock-domain-based connections at the top level. If the relation does not exist in the block level CTL model, then you must specify appropriate pins on the block, leaving the CTL model to do the connections.

If the pins/ports on the CTL modeled cells are specified, then the physical connectivity inside the CTL modeled cell is not validated, and you must determine the valid polarity/active states of the pins/ports on the CTL model cells and specify connectivity appropriately.

If **-target** is not specified, then the tool selects appropriate design elements based on **-type** in the current design.

Pure combinational cells and CTL modeled cells are ignored and their specifications are discarded.

-rise_target *rise_clock_list*

Specifies a list of clocks. All scan cells triggered by the rising edge of the clock are connected to the specified DFT signal driver port or pin.

This option is valid only with **-type scan_enable**. Clocks specified in the *rise_clock_list* must be defined as a valid DFT signal clock with **set_dft_signal**.

-fall_target *fall_clock_list*

Specifies a list of clocks. All scan cells triggered by the falling edge of the clock are connected to the specified DFT signal driver port or pin.

This option is valid only with **-type scan_enable**. Clocks specified in the *fall_clock_list* must be defined as a valid DFT signal clock with **set_dft_signal**.

-exclude *cell_list*

Specifies a list of cells to be excluded from the list of cells that will be connected to the specified driver port or pin.

DESCRIPTION

The **set_dft_connect** command specifies a DFT connectivity association. It is used to connect a DFT signal driver port or pin to a list of specified target cells when the **insert_dft** command is run.

Note that this command has been deprecated; you should use the **set_dft_signal-connect_to** command instead.

Connections are done sequentially. First, connections are done for **-type clock_gating_control** and then for **-type scan_enable**. For each type, connections are again done sequentially, so that subsequent **set_dft_connect** commands take precedence over earlier commands. You must maintain order and precedence by specifying the **set_dft_connect** commands in the correct order.

Use the **report_dft_connect** command to display specified connectivity associations.

Use the **remove_dft_connect** command to remove specified connectivity associations.

EXAMPLES

The following example first sets the signals for ports *SE1* through *SE7*, and then connects the sources to the targets:

```
prompt> set_dft_signal -view spec -type ScanEnable -port SE1 \
        -usage clock_gating

prompt> set_dft_signal -view spec -type ScanEnable -port SE2 \
        -usage clock_gating

prompt> set_dft_signal -view spec -type ScanEnable -port SE3 \
        \fb-usage clock_gating

prompt> set_dft_signal -view spec -type ScanEnable -port SE4 -usage scan

prompt> set_dft_signal -view spec -type ScanEnable -port SE5 -usage scan

prompt> set_dft_signal -view spec -type ScanEnable -port SE6 -usage scan

prompt> set_dft_signal -view spec -type ScanEnable -port SE7 -usage scan

prompt> set_dft_connect label1 -source SE1 -type clock_gating_control \
        -target [list U1 U2]

prompt> set_dft_connect label2 -source SE2 -type clock_gating_control \
        -target [list U1/clock_gate*] -exclude U1/clock_gate_1

prompt> create_clock -p 5 clk1 -name CLK1

prompt> set_dft_signal -view exist -type ScanClock -port clk1 \
        -timing {45 55}

prompt> set_dft_connect label3 -source SE3 -type clock_gating_control \
        -target [list CLK1]

prompt> set_dft_connect label4 -source SE4 -type scan_enable \
        -target [list U1]

prompt> set_dft_connect label5 -source SE4 -type scan_enable \
        -target [list U2] -exclude U2/ff1

prompt> set_dft_signal -view exist -type ScanClock -port clk2 \
        -timing {45 55}
```

```
prompt> set_dft_signal -view exist -type ScanClock -port clk3 \
-timing {45 55}
```

```
prompt> set_dft_signal -view exist -type ScanClock -port clk4 \
-timing {45 55}
```

```
prompt> set_dft_connect label6 -source SE5 -type scan_enable \
-target [list clk2]
```

```
prompt> set_dft_connect label7 -source SE6 -type scan_enable \
-rise_target [list clk2]
```

```
prompt> set_dft_connect label8 -source SE7 -type scan_enable \
-fall_target [list clk3 clk4]
```

The following example connects a dedicated ScanEnable/TestMode on a CTL modeled cell to *SE8*, *SE9*, and *TM* at the top level. The *core* cell has CTL model with *core/SE_scan* and *core/SE_cg* as dedicated ScanEnable signals, and *core/TM_cg* as dedicated TestMode signal.

```
prompt> set_dft_signal -view spec -type ScanEnable -port SE8 \
-usage clock_gating
```

```
prompt> set_dft_signal -view spec -type ScanEnable -port SE9 \
-usage scan
```

```
prompt> set_dft_signal -view spec -type TestMode -port TM \
-usage clock_gating
```

```
prompt> set_dft_connect label9 -source SE8 -type clock_gating_control \
-target core/SE_cg
```

```
prompt> set_dft_connect label10 -source SE9 -type scan_enable \
-target core/SE_scan
```

```
prompt> set_dft_connect label11 -source TM -type clock_gating_control \
-target core/TM_cg
```

SEE ALSO

```
remove_dft_connect(2)
report_dft_connect(2)
set_dft_signal(2)
```

set_dft_drc_configuration

Sets the DFT DRC configuration for the current design.

SYNTAX

```
status set_dft_drc_configuration
  [-internal_pins enable | disable]
  [-pll_bypass enable | disable]
  [-assume_pi_scan enable | disable]
  [-assume_po_scan enable | disable]
  [-static_x_analysis enable | disable]
  [-use_test_model true | false]
  [-clock_gating_init_cycles integer]
  [-allow_se_set_reset_fix true | false]
  [-analyze_multi_clock_activation enable | disable]
```

ARGUMENTS

-internal_pins enable | disable

Enables the "internal pins" flow, in which internal pins can be specified as clock, reset, set, constant, scan-in, scan-out, scan-enable, and test_mode signals.

When **-internal_pins** is set to **enable**, you can specify internal pins to be as clock, reset, set, constant, scan-in, scan-out, scan-enable and test_mode signals. You can specify these internal pins using the **set_dft_signal** and **set_scan_path** commands. These signals are then used by the **dft_drc**, **preview_dft**, and **insert_dft** commands. However, the protocol generated at the end of post-DFT DRC is not complete and cannot be used by the user.

-pll_bypass enable | disable

Enables the PLL bypass mode during post-DFT DRC.

-assume_pi_scan enable | disable

Applies only when BIST is enabled. When this option is enabled (the default), Design Rule Checking does not check that all primary inputs are driven by scan elements, and just assumes that this is true. If you want Design Rule Checking to check that all primary inputs are driven by scan elements, set this option to **disable**. Design Rule Checking then reports L-24 violations on the primary inputs that are not driven by scan elements.

-assume_po_scan enable | disable

Applies only when BIST is enabled. When this option is enabled (the default), Design Rule Checking does not check that all primary outputs drive scan elements, and just assumes that this is true. If you want Design Rule Checking to check that all primary outputs drive scan elements, set this option to **disable**.

-static_x_analysis enable | disable

Enables pre-DFT DRC to report static-X scan cell violations. The default is **disable**.

-use_test_model true | false

Replaces DFT-inserted blocks with their test model representations during DFT DRC and scan architecting. The default is **true**.

When this option is set to **false**, **dft_drc** uses the gates instead of the test model attached to a block. Use this only for blocks containing scan chains and complex test control structures, where the test model does not accurately describe the test control logic.

-clock_gating_init_cycles integer

Adds the specified number of additional clock pulses to the test_setup section of the test protocol. This is useful for initialization of multiple cascaded sequential clock-gating cells. The default is zero.

-allow_se_set_reset_fix true | false

Controls whether pre-DFT and post-DFT DRC allow internally generated reset signals that are gated by the scan-enable signal. When set to the default of **false**, DFT DRC does not allow scan cells to be driven by any internally generated reset; they are marked as violations and excluded from scan chains. When set to **true**, DFT DRC allows scan cells whose internally generated reset signals are gated by the scan-enable signal during scan shift.

Note that scan-enable signals driving existing testability logic in your design must be defined with the **-view existing_dft** option of the **set_dft_signal** command for pre-DFT DRC to consider them, even if they are already also defined with the **-view spec** option.

This option is equivalent to the **set_drc -allow_unstable_set_reset** option in TetraMAX ATPG.

-analyze_multi_clock_activation enable | disable

Enables DFT DRC to analyze scan elements for multiple clock/reset/set active inputs. The D18 violation is reported on scan elements with such inputs. The default is **disable**.

DESCRIPTION

This command specifies the configuration for DFT DRC, which is performed explicitly by the **dft_drc** command and can be performed implicitly by the **preview_dft** and **insert_dft** commands if not previously run.

EXAMPLES

The following example enables the internal pins flow:

```
prompt> set_dft_drc_configuration -internal_pins enable
```

The following example applies additional initialization clock pulses:

```
prompt> set_dft_drc_configuration -clock_gating_init_cycles N \
    ;# where, N = number of clock pulses
```

SEE ALSO

```
set_dft_configuration(2)
set_dft_signal(2)
set_scan_configuration(2)
set_scan_path(2)
write_test_protocol(2)
```

set_dft_drc_rules

Alters how certain DRC rule violations affect DFT insertion for the specified cells.

SYNTAX

```
int set_dft_drc_rules  
  [-allow drc_list]  
  [-ignore drc_list]  
  [-cell cell_list]
```

Data Types

drc_list string
cell_list string

ARGUMENTS

-allow *drc_list*

Specifies that for the specified DRC violations, DFT should allow violating cells to be included in scan chains. The violations are still reported.

-ignore *drc_list*

Specifies that for the specified DRC violations, DFT should completely ignore the violations. The violating cells are included in scan chains and the violations are not reported.

-cell *cell_list*

Limits the DFT DRC specification to certain cells. If no cells are specified, the specification applies globally to all cells.

DESCRIPTION

The **set_dft_drc_rules** command can be used to change how certain DRC rule violations affect DFT insertion. The list of DRC rule violation IDs supported by this command is:

- TEST-504 (warning) Cell %s has constant 0 value.
- TEST-505 (warning) Cell %s has constant 1 value.
- D17 (warning) D17 Clock input I of <type> S couldn't capture data.

By default, these DRC violations cause DFT to omit the violating cells from scan chains. You can use this command to include them in scan chains, with or without the violations being reported.

The **set_dft_drc_rules** command in DFT Compiler is similar to the **set_rules** command in TetraMAX in that both commands control how the tool treats DRC rule violations. However, they are different in that the **set_dft_drc_rules** command controls how the DRC violations affect DFT insertion, while the **set_rules** command controls the severity of DRC violations for a static design.

Note that the following commands take precedence over a **set_dft_drc_rules** specification:

- If a flip-flop also has a **set_scan_element false** attribute applied, it takes precedence, and the **set_dft_drc_rules** specification has no effect.
- If a flip-flop holds a constant value due to the **set_test_assume** command, this assumed constant value takes precedence, and the **set_dft_drc_rules** specification has no effect.

Use the **report_dft_drc_rules** command to display the DFT DRC rule specifications that have been applied. Use the **reset_dft_drc_rules** command to reset all DFT DRC rule specifications.

EXAMPLES

The following example allows all cells that violate TEST-504 and TEST-505 to be included on scan chains, with the violations reported:

```
prompt> set_dft_drc_rules -allow {TEST-504 TEST-505}
```

The following example allows all cells that violate TEST-504 and TEST-505 to be included on scan chains in the hierarchical instance USPARE_GATES, with the violations not reported:

```
prompt> set_dft_drc_rules -ignore {TEST-504 TEST-505} -cell USPAREGATES
```

SEE ALSO

dft_drc(2)
insert_dft(2)
preview_dft(2)
report_dft_drc_rules(2)
reset_dft_drc_rules(2)

set_dft_equivalent_signals

Sets a given list of DFT signals as equivalents.

SYNTAX

```
integer set_dft_equivalent_signals  
    signal_list
```

Data Types

signal_list list

ARGUMENTS

signal_list

Lists the signals to check against the primary signal for equivalency requirements. The first signal in the list is the primary signal. The primary signal is used to check for valid equivalences with the remaining signals in the list for type and other (if any) equivalency requirements. All the signals must have the same signal type.

DESCRIPTION

This command sets a specified list of scan signals as equivalent for scan architect. This is supported only for clock signal types (ScanClock, MasterClock, and ScanMasterClock), as specified by the **set_dft_signal** command or inferred by the **create_test_protocol** command.

Use this command to specify a set of clocks as equivalent for scan architect. This enables the architect to consider all flops driven by these clocks as if driven by the same clock, and the architect will not add any synchronization elements when they are mixed in a scan chain.

EXAMPLES

The following example illustrates setting two clocks as equivalent to clock clk1:

```
prompt> set_dft_equivalent_signals [list clk1 clk2 clk3]
```


SEE ALSO

insert_dft(2)
preview_dft(2)
remove_dft_equivalent_signals(2)
report_dft_equivalent_signals(2)
set_dft_signal(2)

set_dft_insertion_configuration

Sets the DFT insertion configuration for the current design.

SYNTAX

```
int set_dft_insertion_configuration
  [-map_effort low | medium | high]
  [-synthesis_optimization none | all]
  [-route_scan_enable true | false]
  [-route_scan_clock true | false]
  [-route_scan_serial true | false]
  [-preserve_design_name true | false]
  [-unscan true | false]
```

ARGUMENTS

-map_effort low | medium | high

Specifies the optimization effort level during DFT insertion. Valid values are **low**, **medium**, and **high**.

The default is **medium**.

-synthesis_optimization none | all

Specifies whether to perform optimizations during DFT insertion.

In wireload mode, valid values are **all** to turn on all optimizations, and **none** to turn off all optimizations. The default is **all**.

In topographical mode, the only allowed value is **none**; it cannot be changed.

-route_scan_enable true | false

Specifies whether to route scan enables during DFT insertion. Valid values are **true** to perform scan enable routing, and **false** to turn off scan enable routing.

The default is **true**.

-route_scan_clock true | false

Specifies whether to route scan clocks during DFT insertion. Valid values are **true** to route the scan clocks, and **false** to turn off scan clock routing.

The default is **true**.

-route_scan_serial true | false

Specifies whether to route scan serial signals during DFT insertion. Valid values are **true** to route the scan serial signals, and

false to turn off scan serial routing.

The default is **true**.

-preserve_design_name true | false

Specifies whether to preserve the design name when writing from the tool to the database during DFT insertion. Valid values are **true** to preserve the design name, and **false** to permit renaming the design.

The default is **false**.

-unscan true | false

Specifies whether to unscan invalid scan-replaced cells during DFT insertion.

A scan-replaced cell is invalid if it has a DRC violation or if it has the **scan_element** attribute set to **false**.

When set to **false** (the default), invalid scan-replaced cells remain scan-replaced despite not being included in scan chains. However, in wire load mode, if the **-synthesis_optimization** option of this command is set to its default of **all**, cells with the **scan_element** attribute set to **false** are still unscanned.

When set to **true**, invalid scan-replaced cells are unscanned. If an invalid scan-replaced cell has a **dont_touch** attribute, the **dont_touch** takes precedence and this option has no effect.

The **-unscan** option affects only the **insert_dft** command; it does not affect post-DFT optimization (implicit or explicit).

DESCRIPTION

The **set_dft_insertion_configuration** command specifies the configuration to use for the current design during the DFT insertion process. The settings apply only to the current design. If you change the DFT insertion configuration and then change the current design, your changes are no longer visible.

When **-synthesis_optimization none** is specified, the **insert_dft** command does not perform any timing or design rule optimization. In addition, scan cells with the **scan_element** attribute set to **false** are not unscanned.

When **-map_effort low** is specified, the **insert_dft** command resizes only the logic it added to the design.

When **-map_effort medium** is specified, **insert_dft** first resizes the logic it added. If constraints are still violated, it applies a set of heuristic optimizations to all critical points in the design. The heuristics include:

1. *Sizing*: Size one or more drivers or loads for appropriate drive strength.
2. *Phasing*: Replace the driver with one that has an opposing phase.
3. *Buffering*: Insert buffers or inverter pairs; transform inverter pairs to buffers, or transform buffers to inverter pairs or buffer inverters.
4. *Downsizing*: Downsize and swap out loads.
5. *Isolation*: Isolate noncritical loads using inverter pairs or buffers.
6. *Off-loading*: Offload noncritical loads to other driver outputs, use a driver output with a greater drive strength, use drivers with more outputs, or move loads to nets with earlier arrival times.
7. *Balancing*: Redistribute loads among drivers, balance buffers, or balance loads across driver outputs.
8. *Splitting*: Duplicate driver, distributing loads or dedicating one driver to the critical path.

When **-map_effort high** is specified, the **insert_dft** command first resizes the logic it added. If constraints are still violated, it performs critical-path optimization on the entire design. If constraints are still violated, it applies sequential mapping to sequential elements on critical paths. It then attempts to reduce design area by downsizing cells that are not on a critical path and eliminating extra inverters. It attempts to eliminate any remaining constraint violations by using normal critical-path optimization.

Use the **report_dft_insertion_configuration** command to display the current DFT insertion configuration of the design.

Use the **reset_dft_insertion_configuration** command to remove the current DFT insertion configuration from the design.

EXAMPLES

The following command sets the mapping effort level to **high**:

```
prompt> set_dft_insertion_configuration -map_effort high
```

The following command turns off all synthesis optimizations in Design Compiler wireload mode:

```
prompt> set_dft_insertion_configuration \  
-synthesis_optimization none
```

SEE ALSO

insert_dft(2)
report_dft_insertion_configuration(2)
reset_dft_insertion_configuration(2)

set_dft_location

Specifies the DFT hierarchy location for DFT insertion.

SYNTAX

status **set_dft_location**

dft_hier_name

[**-exclude** CODEC | TCM | LOCKUP_LATCH | PIPELINE_SI_LOGIC | PIPELINE_SE_LOGIC | WRAPPER | BSR | TAP | REC_MUX |
[**-include** CODEC | TCM | LOCKUP_LATCH | PIPELINE_SI_LOGIC | PIPELINE_SE_LOGIC | WRAPPER | BSR | TAP | REC_MUX |

Data Types

dft_hier_name string

ARGUMENTS

dft_hier_name

Specifies the hierarchical path name of an instance in the current design into which all DFT logic will be added during DFT insertion.

This argument is required.

-exclude CODEC | TCM | LOCKUP_LATCH | PIPELINE_SI_LOGIC | PIPELINE_SE_LOGIC | WRAPPER | BSR | TAP | REC_MUX |
PLL | SERIAL_CNTRL | SERIAL_REG | XOR_SELECT | RETIMING_FLOP | IEEE_1500

Specifies that all types of DFT logic except the specified types are to be included in the specified DFT location.

Note that this option overwrites any previous DFT location specifications for DFT logic types that are not specified. See the final **-exclude** example in the EXAMPLES section for more information.

More than one type of DFT logic can be excluded with this option.

There is no default for this option. When this option is not specified, the location specification applies as a default to all DFT logic types.

-include CODEC | TCM | LOCKUP_LATCH | PIPELINE_SI_LOGIC | PIPELINE_SE_LOGIC | WRAPPER | BSR | TAP | REC_MUX |
PLL | SERIAL_CNTRL | SERIAL_REG | XOR_SELECT | RETIMING_FLOP | IEEE_1500

Specifies the types of DFT logic to be included in the specified DFT location.

CODEC logic includes the compressor and decompressor (codec) logic inserted by the tool for compressed scan, serialized compressed scan, and streaming compressed scan modes.

PIPELINE_SI_LOGIC includes all head and tail pipelined scan data registers.

PIPELINE_SE_LOGIC includes all pipelined scan-enable logic.

PLL logic includes PLL controller and clock chain logic.

WRAPPER logic includes dedicated wrapper cells and wrapper mode logic.

BSR logic includes all BSR cells and instruction decode and mode logic.

TAP logic includes the TAP controller instance.

SERIAL_CNTRL logic includes the serializer clock controller used in serializer insertion flows.

SERIAL_REG logic includes the serializer and deserializer registers used in the serializer IP insertion flow. This type does not affect the normal serializer insertion flow.

XOR_SELECT includes the sharing compressor and block-select logic inserted at the compressor outputs in the shared codec I/O flow.

TCM logic includes the decoder and mode logic.

LOCKUP_LATCH logic includes all lock-up latches.

RETIMING_FLOP logic includes all retiming flip-flops.

REC_MUX logic includes scan-reconfiguration muxes, scan-out muxes, tri-state/bidir disable logic, and any other glue logic added during DFT insertion.

IEEE 15000 logic includes all DFT-inserted IEEE 1500 logic used for test-mode control. At the chip level, it also includes the server logic that interfaces to the IEEE Std 1149.1 TAP controller.

More than one type of DFT logic can be included with this option.

There is no default for this option. When this option is not specified, the location specification applies as a default to all DFT logic types.

DESCRIPTION

The **set_dft_location** command can be used to add all DFT logic into a user-specified hierarchy during DFT insertion.

If the **-include** and **-exclude** options are not specified, the location specification applies as a default to all DFT logic types. Any type-specific specifications take precedence over this default location specification. You can use the **report_dft_location** command to report the resulting location specification.

In the normal serializer insertion flow, the deserializer and serializer registers are always placed inside the decompressor and compressor, respectively, whose location is specified with the CODEC logic type. To place the deserializer and serializer registers in a different hierarchy location than the combinational codec logic, use the serializer IP insertion flow and specify the register location with the SERIAL_REG logic type.

EXAMPLES

The following example specifies a DFT hierarchy location for the design top. All types of DFT logic are included in the specified location.

```
prompt> set_dft_location core_inst/dft_inst
Accepted DFT location specification.
```

1

The following example specifies a DFT hierarchy location for the design top. All wrapper and lock-up latch logic should not be inserted at this location.

```
prompt> set_dft_location core_inst/dft_inst \
-exclude { WRAPPER LOCKUP_LATCH }
Accepted DFT location specification.
```

1

The following example specifies a DFT hierarchy location for wrapper logic. The codec logic is inserted at another DFT location. Any other DFT logic should not be inserted in these locations.

```
prompt> set_dft_location core_inst/dft_inst1 -include { WRAPPER }
Accepted DFT location specification.
1
prompt> set_dft_location core_inst/dft_inst2 -include { CODEC }
Accepted DFT location specification.
```

1

The following example specifies a DFT hierarchy location for BSD logic. Both BSR and TAP logic are added to this location.

```
prompt> set_dft_location core_inst/bsd_inst -include { BSR TAP }
Accepted DFT location specification.
```

1

The following example specifies a DFT hierarchy location for BSR logic. The TAP logic is inserted at another DFT location.

```
prompt> set_dft_location core_inst/bsd_inst1 -include { BSR }
Accepted DFT location specification.
1
prompt> set_dft_location core_inst/bsd_inst2 -include { TAP }
Accepted DFT location specification.
```

1

The following example specifies a DFT hierarchy location for CODEC logic, then sets a DFT hierarchy location for everything except lock-up latches. Because the **-exclude** option includes all DFT logic types except the specified type(s), the CODEC specification is overwritten.

```
prompt> set_dft_location UCODEC -include { CODEC }
Accepted DFT location specification.
1
prompt> set_dft_location UDFT -exclude { LOCKUP_LATCH }
Accepted DFT location specification.
```

1

```
prompt> report_dft_location
```

Design Name	DFT PARTITION NAME	DFT TYPE	DFT Hierarchy Location

top	default_partition	BSR	UDFT
top	default_partition	CODEC	UDFT
top	default_partition	LOCKUP_LATCH	<local>
top	default_partition	XOR_SELECT	UDFT
top	default_partition	RETIMING_FLOP	UDFT
top	default_partition	TAP	UDFT
top	default_partition	TCM	UDFT
top	default_partition	PIPELINE_SE_LOGIC	UDFT
top	default_partition	PIPELINE_SI_LOGIC	UDFT
top	default_partition	PLL	UDFT
top	default_partition	REC_MUX	UDFT

top	default_partition	SERIAL_CNTRL	UDFT
top	default_partition	SERIAL_REG	UDFT
top	default_partition	WRAPPER	UDFT
top	default_partition	IEEE_1500	UDFT

SEE ALSO

insert_dft(2)
remove_dft_location(2)
report_dft_location(2)

set_dft_power_control

Specifies the power controller block instance for the current design.

SYNTAX

```
status set_dft_power_control  
      power_controller_hierarchical_instance_name
```

ARGUMENTS

power_controller_hierarchical_instance_name

Specifies the power controller hierarchical instance name.

DESCRIPTION

This command allows you to identify the power controller block instance for the power control override insertion. DFT insertion then inserts a dedicated wrapper chain at the output pins to control power management structures such as power switches, isolation cells and retention registers during test. The power control signals at the block output pins are defined using commands that are part of the IEEE 1801 specification, also known as the Unified Power Format (UPF) specification.

You can only have one power controller block specified for the current design. Subsequent commands will override the previous specification.

Use the **report_dft_power_control** command to report the current power controller block specification. Use the **remove_dft_power_control** command to remove the current power controller block specification.

EXAMPLES

The following command specifies an instance of power controller instance.

```
prompt> set_dft_power_control U_PWC
```

SEE ALSO

```
remove_dft_power_control(2)
report_dft_power_control(2)
set_dft_configuration(2)
```

set_dft_signal

Specifies the DFT signal types for DRC and DFT insertion.

SYNTAX

```
status set_dft_signal
[-view existing_dft | spec]
[-test_mode mode_name_list]
-type signal_type
[-port port_list]
[-active_state 0 | 1]
[-timing timing]
[-period period]
[-hookup_pin hookup_pin]
[-hookup_sense hookup_sense]
[-internal_clocks none | single | multi]
[-ctrl_bits ctrl_bits_list]
[-pll_clock clock_name]
[-ate_clock clock_name]
[-differential_clock clock_port]
[-connect_to object_list]
[-connect_to_domain_rise clock_list]
[-connect_to_domain_fall clock_list]
[-usage use_type]
[-associated_internal_clocks clock_pins_list]
[-associated_clock associated_clock]
[-exclude cell_list]
[-codec decompressor_name]
```

Data Types

```
mode_name_list    string
signal_type      string
port_list        list
timing           list
period          float
hookup_pin       list
hookup_sense     string
ctrl_bits_list   list
clock_name       string
clock_port       string
object_list      list
clock_list       list
use_type         list
associated_clock string
cell_list        list
decompressor_name string
```

ARGUMENTS

-view existing_dft | spec

Indicates the view to which the specification applies. The following views are valid:

- **existing_dft** implies that the specification refers to the existing usage of a port. For example, when working with a design that is already DFT-inserted, the command to indicate that port SE is used as a scan-enable port is as follows:

```
set_dft_signal -view existing_dft -port SE -type ScanEnable
```

- **spec** (the default) implies that the specification refers to ports that the tool must use during DFT insertion. For example, when preparing a design for DFT insertion, the command to specify that port SE is used as a scan-enable port for DFT insertion is as follows:

```
set_dft_signal -view spec -port A -type ScanEnable
```

-test_mode mode_name_list

Specifies the test mode(s) to which the specification applies. Test modes are created with the **define_test_mode** command. A value of **all** or **all_dft** causes the specification to apply to all test modes.

If the **-test_mode** option is not specified, the command applies to the current test mode, which is determined by the last **define_test_mode** or **current_test_mode** command executed. If no test modes have been defined, the command applies to the default test mode.

-type signal_type

Specifies the signal type. The valid signal types are as follows:

- **Reset** is the asynchronous set or reset of the design.
- **Constant** is a continuously applied value to a port.
- **TestMode** is a continuously applied value to a test mode port that may have different values between test modes.
- **TestData** is the port used to apply scan data to a portion of the design that has had pre-DRC violations fixed by Autofix.
- **TestControl** is the port used to control the DFTMAX shift power groups feature.
- **ScanDataIn** is one of the scan inputs of the design.
- **ScanDataOut** is one of the scan outputs of the design.
- **ScanEnable** is the shift enable of a MUX-D scan design.
- **ScanClock** is a clock that performs both scan shift and capture in a MUX-D scan design. (This type is equivalent to defining the clock as both the MasterClock and ScanMasterClock types; there is no CTL signal type called ScanClock.)
- **ScanMasterClock** is the clock responsible for scan shift. In an LSSD scan style, the ScanMasterClock is the A clock, and in a clocked scan style it is the shift clock.
- **ScanSlaveClock** is the B clock in an LSSD scan style.
- **MasterClock** is the clock responsible for capture. In an LSSD scan style the MasterClock is the system clock, and in a clocked scan style it is the capture clock.

In general, the MasterClock is used for referring to the capture clock and ScanMasterClock for referring to the shift clock. In the

LSSD scan style, the two shift clocks are specified as ScanMasterClock and ScanSlaveClock, and only the capture clock is specified as MasterClock.

In the MUX-D scan style, the same clock provides both shift and capture functionality. In this case, use the ScanClock signal type, which simultaneously specifies both MasterClock and ScanMasterClock types. To create a shift-only clock in the MUX-D scan style, use the ScanClock type, then apply the **add_pi_constraint** command in TetraMAX.

To specify the sense for an internal hookup-only pin (no port is associated with the hookup pin), use the **-active_state** option.

The following signal types are used in on-chip clocking (OCC) controller flows:

- **Oscillator** specifies the output pin of the on-chip clock generator or the output pins of the user-instantiated on-chip clock controller. In both cases, the **-hookup_pin** option must be used. This is a constantly running clock signal that never turns off.
- **RefClock** is a constantly running external reference clock. The clock can have timing which differs from the test default period.
- **pll_reset** is the asynchronous signal resetting the OCC logic.
- **pll_bypass** is the signal making the OCC logic transparent when the active value is applied.

The following signal types can be used in core-wrapping flows:

- **wrp_shift** specifies the scan shift signal for wrapper cells.
- **wrp_clock** specifies the clock signal for dedicated wrapper cells.
- **input_wrp_shift** specifies the scan shift signal for input wrapper cells in the maximized reuse flow. It can be used only with the **-connect_to** option.
- **output_wrp_shift** specifies the scan shift signal for output wrapper cells in the maximized reuse flow. It can be used only with the **-connect_to** option.
- **wrp_ded_capture_in** specifies the input capture signal in the maximized reuse flow. It is used for input dedicated wrapper cells when creating a wrapped core for hierarchical wrapping flows. It is also used for input shared wrapper cells in custom core wrapping flows.
- **wrp_ded_capture_out** specifies the output capture signal in the maximized reuse flow. It is used for output dedicated wrapper cells when creating a wrapped core for hierarchical wrapping flows. It is also used for output shared wrapper cells in custom core wrapping flows.
- **wrp_ded_capture_inout** specifies the input/output capture signal in the maximized reuse flow. It is used only for shared wrapper cells which are connected to both input and output ports in custom core wrapping flows.
- **wrp_safe_in** specifies the safe control signal for input wrapper cells in the maximized reuse flow. This signal is used when creating a wrapped core for hierarchical wrapping flows.
- **wrp_safe_out** specifies the safe control signal for output wrapper cells in the maximized reuse flow. This signal is used when creating a wrapped core for hierarchical wrapping flows.
- **wrp_td_test** specifies the signal used in core wrapping to aid transistion delay testing. The signal controls the hold state of the wrapper cell in capture. The cell can either hold the state or toggle in capture based on this signal. This signal is used only in custom core wrapping flows.

The following signal types can be used to define IEEE 1500 controller signals in IEEE 1500 flows:

- **WSI** specifies the scan-in signal.
- **WRSTN** specifies the reset signal.
- **WRCK** specifies the clock signal.

- **CaptureWR** specifies the capture-enable signal.
- **ShiftWR** specifies the shift-enable signal.
- **UpdateWR** specifies the update-enable signal.
- **SelectWIR** specifies the shift-path selection signal, which selects between the instruction register and data registers.
- **WSO** specifies the scan-out signal.

The following signal types can be used to define LogicBIST signals in LogicBIST self-test flows:

- **lbistEnable** specifies the signal that enables self-test mode. This also enables any wrapper or test-point logic associated with the self-test mode.
- **lbistStart** specifies the signal that begins self-test operation.
- **lbistStatus_1** and **lbistStatus_0** specify: 00 when idle; 01 when running; 10 when test complete and passed; and 11 when test complete and failed.
- **lbistSeedValue** specifies signals of a bus that provides the BIST seed value.
- **lbistSignatureValue** specifies signals of a bus that provides the BIST signature value.
- **lbistPatternCount** specifies signals of a bus that provides the pattern count used for BIST.
- **lbistShiftLength** specifies the signals of a bus that provide the number of shift cycles per pattern for each BIST pattern.
- **lbistBurnInEnable** specifies the signal that enables burn-in operation.
- **lbistBurnInStopOnFail** specifies whether burn-in operation should stop or continue if self-test fails.
- **lbistCaptureCycleEnable** specifies the signals of a bus that provide the number of capture cycles used for BIST.

The following signal type can be used in the shared codec I/O flow:

- **codec_enable** specifies a signal that enables a codec when the shared controls feature is used. It must be driven by a hookup pin; port-driven signals are not supported.

The following signal types can be used to define PCO wrapper chain signals in the power controller override (PCO) flow:

- **pco_shift_clk** specifies the shift clock signal.
- **pco_update_clk** specifies the update clock signal.
- **pco_override** specifies the override signal.

The following signal types can be used in BSD synthesis and integration:

- **tdi** specifies the TDI port or bsd reg tdi access pin.
- **tdo** specifies the TDO port or bsd reg tdo access pin.
- **tdo_en** specifies the TDO port enable pin.
- **tck** specifies the TCK port.
- **tms** specifies the TMS port.
- **trst** specifies the TRST port.
- **bsd_shift_en** specifies the register access pin to be hooked up to TAP shift_dr pin when the instruction that selects the register is active.

- **bsd_capture_en** specifies the register access pin to be hooked up to TAP sync_capture_en pin when the instruction that selects the register is active. This signal is also used in core integration.
- **bsd_capture_dr** specifies the register access pin to be hooked up to TAP Capture-DR state on the negative edge of TCK when the instruction that selects the register is active.
- **bsd_update_en** specifies the register access pin to be hooked up to TAP sync_update_dr pin when the instruction that selects the register is active. This signal is also used in core integration.
- **bsd_update_dr** specifies the register access pin to be hooked up to TAP Update-DR state on the negative edge of TCK when the instruction that selects the register is active.
- **capture_clk** specifies the register access pin to be hooked up to TCK/clock_dr pin when the instruction that selects the register is active.
- **update_clk** specifies the register access pin to be hooked up to TCK/update_dr pin when the instruction that selects the register is active.
- **inst_enable** specifies the register access pin to be held active when the instruction that selects the register is active.
- **bist_enable** specifies the register access pin to be held active when the instruction that selects the register is active and TAP is in Run-Test-Idle state.
- **bist_clk** specifies the register access pin to be hooked up to TCK when the instruction that selects the register is active and TAP is in Shift-DR state. This signal is also used in core integration.
- **bsd_reset** specifies the register access pin to be hooked up to the Test-Logic-Reset FSM state signal when the instruction that selects the register is active. When the bsd_reset signal is not part of any instruction's test data register definition, it is hooked up directly to the Test-Logic-Reset state signal.
- **bsd_test_logic_reset** specifies the Test-Logic-Reset TAP FSM state signal.
- **bsd_run_test_idle** specifies the Run-Test-Idle TAP FSM state signal.
- **bsd_select_dr_scan** specifies the Select-DR TAP FSM state signal.
- **bsd_capture_dr** specifies the Capture-DR TAP FSM state signal.
- **bsd_shift_dr** specifies the Shift-DR TAP FSM state signal.
- **bsd_exit1_dr** specifies the Exit1-DR TAP FSM state signal.
- **bsd_pause_dr** specifies the Pause-DR TAP FSM state signal.
- **bsd_exit2_dr** specifies the Exit2-DR TAP FSM state signal.
- **bsd_update_dr** specifies the Update-DR TAP FSM state signal.
- **bsd_select_ir_scan** specifies the Select-IR TAP FSM state signal.
- **bsd_capture_ir** specifies the Capture-IR TAP FSM state signal.
- **bsd_shift_ir** specifies the Shift-IR TAP FSM state signal.
- **bsd_exit1_ir** specifies the Exit1-IR TAP FSM state signal.
- **bsd_pause_ir** specifies the Pause-IR TAP FSM state signal.
- **bsd_exit2_ir** specifies the Exit2-IR TAP FSM state signal.
- **bsd_update_ir** specifies the Update-IR TAP FSM state signal.
- **bsd_mode_in** is the mode signal for input BSR cells.

- **bsd_mode_out** is the mode signal for output BSR cells.
- **bsd_mode1_inout** is the mode signal for mode1 of BC_7 BSR cells.
- **bsd_mode2_inout** is the mode signal for mode2 of BC_7 BSR cells.

-port *port_list*

Indicates the list of ports on which to apply the specifications.

-active_state 0 | 1

Specifies the active states for the following signal types:

ScanEnable
Reset
Constant
TestMode
pll_reset
pll_bypass

The active state can be 0 or 1 and it specifies the active sense of the port (high or low) or an internal hookup only pin (no port is associated with the hookup pin).

Wildcards and collections are supported.

-timing *timing*

Specifies the rise time and fall time for clocks. For example:

```
set_dft_signal -view existing_dft -type ScanClock \
-port CLK -timing [list 45 55]
```

-period *period*

Specifies the period of the on-chip clocking (OCC) reference clock. The period value is a floating point number in nanoseconds. When **-period** is used, the values specified after **-timing** indicate the leading edge and the trailing edge of the reference clock. The **-period** option is valid only on signals of the RefClock type.

-hookup_pin *hookup_pin*

Specifies the pin to which signals are to be connected. By default, the **insert_dft** command connects wires to the core side of identified signal ports, jumping pads and buffers as needed. The **-hookup_pin** argument overrides this behavior and instructs **insert_dft** to connect wires to the specified pin. The pin can be either a leaf pin or hierarchical pin. Validation ensures that the pin direction (driver or load) is consistent with the signal type. Verify that a specific access pin is associated with only one design port.

If hookup pins are specified, only one port can be specified in the *port_list*.

However, if the internal pins flow is enabled (**set_dft_drc_configuration -internal_pins**), a hookup pin can be specified without specifying a port. The hookup pin specified should be either on a black box or should have a proper global driver. If a proper global driver is not found, **insert_dft** will not stitch to the specified hookup pin.

-hookup_sense *hookup_sense*

Specifies the hookup sense for the hookup pin. This option is valid only when a single hookup pin or a port is specified. The valid values are **inverted** and **non_inverted**. The default is **non_inverted**.

-internal_clocks none | single | multi

Specifies the setting for an internal clock. An internal clock is defined as an internal signal driven by a multiplexer (or multiple input gate) output pin. This option applies only to the multiplexed flip-flop scan style and it is ignored for other scan styles.

Note that the output of clock gating cells that are introduced by the Synopsys Power Compiler will not be treated as internal clocks, even if the **-internal_clocks** option is enabled.

The **-internal_clocks** option can be set to the following values:

- **single** specifies that **insert_dft** treats any internal clocks in the design as separate clocks for the purpose of scan chain architecting. The **single** value instructs the tool to stop at the first buffer or inverter driving the flip-flops clock.
- **none** (the default) specifies that **insert_dft** does not treat internal clocks as separate clocks.
- **multi** specifies that **insert_dft** treats any internal clocks in the design as separate clocks for the purpose of scan chain architecting. The **multi** value instructs the tool to jump over buffers and inverters, stopping at the first multi-input gate driving the flip-flops clock.

-ctrl_bits *ctrl_bits_list*

Lists triplets that specify the sequence of control bits needed to enable the propagation of the clock generator outputs. This option is used in the on-chip clocking (OCC) flow. The first element of each triplet is the cycle number (integer) indicating when the clock signal will be propagated. The second element is the pin name of the control bit (a valid design hierarchical pin name). The third element is the active value (0 or 1) of the control bit. For example:

```
set_dft_signal -type Oscillator -hookup_pin pll_controller/clk \
-pll_clock pll_i/pll_clk -ate_clock ate_clk \
-ctrl_bits [list \
0 clk_chain_i/clk_ctrl_data[0] 1 \
1 clk_chain_i/clk_ctrl_data[1] 1 \
2 clk_chain_i/clk_ctrl_data[2] 1] \
-view existing_dft
```

-pll_clock *clock_name*

Specifies the port name of the pll clock. This option is used in the on-chip clocking (OCC) flow.

-ate_clock *clock_name*

Specifies the port name of the ATE clock. This option is used in the on-chip clocking (OCC) flow.

-differential_clock *clock_port*

Specifies the pin name of a differential clock. For example, if *clk_m* is the minus side of *clk_p*:

```
set_dft_signal -view existing_dft -type ScanClock \
-port clk_m -differential {clk_p}
```

Note that the minus side will track all of the properties of the plus side with an inverted polarity. A complete example for *clk_m* and *clk_p* is as follows:

```
set_dft_signal -view existing_dft -type Oscillator \
-port clk_p
set_dft_signal -view existing_dft -type ScanClock \
-port clk_p -timing [list 45 55]
set_dft_signal -view existing_dft -type ScanClock \
-port clk_m -differential {clk_p}
```

-connect_to *object_list*

Specifies that the DFT signal should be connected from the port or hookup pin to only the specified design objects.

You can use the **-connect_to** option to define ScanEnable, TestMode, or wrp_shift signals that should only be connected to a specific set of design objects. These are known as object-specific signal definitions. Depending on the signal type, only certain design object types can be specified and certain signal usages are supported. For the different signal types and usages defined

with the **-type** and **-usage** options, the allowed object types for the **-connect_to** option are:

- **-type ScanDataIn**
 - **pin** (ScanDataIn pins of CTL-modeled cores)
- **-type ScanDataOut**
 - **pin** (ScanDataOut pins of CTL-modeled cores)
- **-type ScanEnable -usage scan**
 - **cell** (leaf scan cells)
 - **cell** (hierarchical cell containing scan cells)
 - **design** (designs containing scan cells)
 - **clock** (scan cells clocked by specified clocks)
 - **pin** (ScanEnable pins of CTL-modeled cores)
- **-type ScanEnable | TestMode -usage clock_gating**
 - **cell** (clock-gating cells)
 - **cell** (hierarchical cell containing clock-gating cells)
 - **design** (designs containing clock-gating cells)
 - **clock** (test clocks gated by clock-gating cells)
 - **pin** (ScanEnable or TestMode pins of CTL-modeled cores)
- **-type wrp_shift | input_wrp_shift | output_wrp_shift**
 - **clock** (wrapper cells clocked by specified clocks)

For the **clock_gating** usage, you can specify ScanEnable or TestMode signal definitions, depending on the type of clock-gating control signal (scan-enable or test-mode) used by the specified objects.

Pins of CTL-modeled cores are not supported when performing simple HSS integration of standard scan cores.

When you specify a clock-domain-based signal specification by specifying clocks, they must be defined as valid test clocks with the **set_dft_signal** command. The propagation of the test clock determines the scope of the specification; the propagation of any underlying functional clocks at the clock source is not considered.

When using clock-domain-based signal specifications with CTL-modeled cores, the following requirements must be observed during core creation to ensure that the CTL models contain clock-domain information:

- Core-level ScanEnable signals must be defined using the **-usage** option of the **set_dft_signal** command. This ensures that the core's internal scan-enable signal is not used outside its intended usage.
- Core-level ScanEnable signals defined with a usage of **clock_gating** must also be defined as domain-specific signals using the **-connect_to clock_list** option of the **set_dft_signal** command. This ensures that clock-specific clock-gating annotations are included in the CTL model.

If the cores do not contain clock-domain information, you can directly specify the ScanEnable pins of the CTL-modeled core instead.

The following example defines two clock-domain-based ScanEnable signals:

```
set_dft_signal -view exist -type ScanClock -timing {45 55} \
```

```
-port {CLK1 CLK2}
```

```
set_dft_signal -type ScanEnable -port SE1 -usage scan \
-connect_to CLK1
set_dft_signal -type ScanEnable -port SE2 -usage scan \
-connect_to CLK2
```

The following example defines a ScanEnable signal to be used only as the clock-gating control signal for a specific block:

```
set_dft_signal -type ScanEnable -port SE_CG_IP -usage clock_gating \
-connect_to UIP_CORE
```

The **-connect_to** option is also used in the hierarchical on-chip clocking (OCC) flow to specify the correspondence between the top-level ATE clock port and the core-level ATE clock pins. For existing top-level connections, use the **-view existing** option. For connections to be made by DFT insertion, use the **-view spec** option. For example,

```
set_dft_signal -view spec -port TOP_CLK -type MasterClock \
-connect_to {CORE1/ATECLK CORE2/ATECLK} -timing {45 55}
```

Clock-domain-based wrapper shift signals are supported only in the maximized reuse flow.

-connect_to_domain_rise *clock_list*

Specifies a clock-domain-based ScanEnable specification that applies only to rising-edge scan cells.

This option works the same as specifying a test clock with the **-connect_to** option, except that it applies only to scan cells clocked by the rising edge of the specified test clocks. It is valid only with **-type ScanEnable** and **-usage scan** options.

This option is not supported for wrapper shift signals.

-connect_to_domain_fall *clock_list*

Specifies a clock-domain-based ScanEnable specification that applies only to falling-edge scan cells.

This option works the same as specifying a test clock with the **-connect_to** option, except that it applies only to scan cells clocked by the falling edge of the specified test clocks. It is valid only with **-type ScanEnable** and **-usage scan** options.

This option is not supported for wrapper shift signals.

-usage *use_type*

Specifies the usage of the signal. You must also specify **-view spec** with this option. This option specifies that **insert_dft** is to use the specified signal for that purpose only.

The signal must be defined with suitable valid values when using it to connect design elements using the **set_dft_connect** command. The valid values for *use_type* are as follows:

- **clock_gating** specifies that the signal is to be connected to the test pin of identified clock gating cells during **insert_dft**. This usage is valid only for **-type ScanEnable** and **-type TestMode**.

If clock-gating test pin connection is disabled by setting the **-connect_clock_gating** option of the **set_dft_configuration** command to **disable**, this signal is not used.

- **scan** specifies that the signal is to be connected to the enable of scan elements during **insert_dft**. This usage is valid only for **-type ScanEnable**.

Multiple usages can be specified for a signal by specifying them as a list. For example, a ScanEnable can be used to connect to test pins of clock gating cells as well as enabling scan elements.

For example,

```
set_dft_signal -view spec -port SE \
```

-type ScanEnable -usage {scan clock_gating}

-associated_internal_clocks *clock_pins_list*

Specifies a list of internal pins associated to be associated with a clock source port.

The tool assumes that the port-driven clock signal is also driven at these internal pins, with the same timing waveform and polarity. The fanout from each internal clock pin is treated as a separate clock domain when architecting scan chains, and the pin names are shown as the clock names by the **preview_dft** and **insert_dft** commands.

Only leaf pins can be specified; hierarchical pins are not supported.

For example,

```
set_dft_signal -type ScanClock -port CLK \
-associated_internal_clocks [list U1/int_clk U2/int_clk1 U1/int_clk2] \
-view existing
```

During pre-DFT DRC, the tool does not check for logical connectivity between the port and internal clock pins. This provides a method to bypass black boxes or other logic with unknown functionality in the clock network.

The connectivity is checked in post-DFT DRC.

-associated_clock *associated_clock*

Specifies the pin or port name of a single clock that you want to associate with a specified scan-enable signal.

For example,

```
set_dft_signal -view spec -type ScanEnable \
-port SE1 -associated_clock CLK1
```

This option is used only in a domain-based scan-enable flow, enabled by setting the **-domain_based_scan_enable** option of the **set_scan_configuration** command to **true**. If domain-based scan-enable signals are not enabled, this option is ignored.

When using this option, an **-associated_clock** specification must be issued for all scan clock domains that use scan-enable signals; otherwise, bad logic might result.

-exclude *cell_list*

Specifies a list of objects to be excluded from an object-specific DFT signal definition made with the **-connect_to**, **-connect_to_domain_rise**, or **-connect_to_domain_fall** option.

In some cases, an object-specific connection specification might require certain objects to be excluded, such as excluding specific leaf cells from a hierarchical block. In this case, you can specify the objects to exclude by using the **-exclude** option. The allowed object types for the **-exclude** option are:

- **-type ScanEnable -usage scan**
 - **cell** (leaf scan cells)
 - **cell** (hierarchical cell containing scan cells)
 - **design** (designs containing scan cells)
- **-type ScanEnable | TestMode -usage clock_gating**
 - **cell** (clock-gating cells)
 - **cell** (hierarchical cell containing clock-gating cells)
 - **design** (designs containing clock-gating cells)

This option excludes the specified objects from the current signal's specification, but it does not prevent those objects from being connected to other signals.

This option can only be used together with the **-connect_to**, **-connect_to_domain_rise**, or **-connect_to_domain_fall** option.

For example:

```
set_dft_signal -type ScanEnable -port SE1 -usage scan \
-connect_to {CLK1} -exclude UIP_BLOCK
```

-codec *decompressor_name*

Specifies the codec to be controlled by a signal defined with the **-type codec_enable** option. Reference the codec's decompressor using the *cell_name:decompressor_name* syntax.

By default, the tool chooses a codec for the signal to control.

DESCRIPTION

The **set_dft_signal** command can be used to specify one or more primary input or output ports as DFT signals. It can be used either to describe the existing usage or to prescribe the usage during implementation.

A DFT signal has a port and a signal type. You can specify multiple DFT signals, but you can specify only one signal type. Port directions must be consistent with the signal type.

You can use the **set_dft_signal** command to declare different DFT signals for different test modes. For example, each test mode can have different scan-in and scan-out ports, or different constant, set, or reset signals. However, all test modes share the same scan-enable signals. You cannot specify different scan-enable signals for different test modes.

The **set_dft_signal** commands are not incremental. A **set_dft_signal** command that identifies a port overwrites any previous **set_dft_signal** command that identifies the same port.

To review your DFT signal specifications, use the **report_dft_signal** command. To remove a specification, use the **remove_dft_signal** command. To implement the DFT signal, use the **insert_dft** command.

Descriptive specification (using **-view existing_dft**) will invalidate test protocol and details about DFT structures.

SEE ALSO

```
current_design(2)
dft_drc(2)
insert_dft(2)
preview_dft(2)
remove_dft_signal(2)
report_dft_signal(2)
set_dft_configuration(2)
set_dft_drc_configuration(2)
write(2)
```

set_direct_power_rail_tie

Sets the **direct_power_rail_tie** attribute on library pins.

SYNTAX

```
int set_direct_power_rail_tie  
    lib_pin_list[true | false]
```

ARGUMENTS

lib_pin_list

A list of library pins on which the **direct_power_rail_tie** attribute is to be set. If more than one library pin name is specified, they must be enclosed either in quotes or in braces ({}). For more information about object names, refer to the **find** command manual page.

true | false

Specifies the value with which to set the **direct_power_rail_tie** attribute. The default is *true*.

DESCRIPTION

Sets the **direct_power_rail_tie** attribute on library pins. If a match is found, a **direct_power_rail_tie** attribute is assigned to the library_pin(s).

If the value of **direct_power_rail_tie** attribute on a library pin is true, then all the pins in the design, for which this is the library pin, it won't connect it to a tieoff cell for connecting it to constant signal. Instead, it will keep them connecting to generic constant signals, so that during power routing these pins can be connected directly to power rails.

To remove **direct_power_rail_tie** attribute, use **remove_attribute**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command specifies that the all the **GND** pins of cells with the library cells **leaf_cell** won't be connected to tieoff cells to

connect it to a constant signal.

prompt> **set_direct_power_rail_tie target_lib/leaf_cell/GND**

SEE ALSO

report_direct_power_rail_tie(2)

remove_attribute(2)

set_disable_clock_gating_check

Disables the clock-gating check for the specified objects in the current design.

SYNTAX

```
status set_disable_clock_gating_check  
      object_list
```

Data Types

object_list collection

ARGUMENTS

object_list

Specifies the cells and pins for which the clock-gating check is to be disabled.

DESCRIPTION

This command disables the clock-gating check for the specified cells and pins in the current design.

Any cell or pin in the current design or its subdesigns can be disabled. Cells at lower levels in the design hierarchy are specified as *instance1/instance2/cell_name*. Specify pins at lower levels in the design hierarchy as *instance1/instance2/cell_name/pin_name*.

For subdesigns in the hierarchy, specify instance names instead of design names.

When the clock-gating checking through a cell is disabled, all gating checks in the cell are disabled. When the checking through a pin is disabled, a gating check is disabled if it uses the disabled pin as a gating clock pin or a gating enable pin.

The **compile** command adjusts only the size of the cell by replacing it with other cells having the same logic function but with different drive capacity. No other logic transformations involving the clock-gating cell are permitted.

To restore the gating checks disabled by the **set_disable_clock_gating_check** command, use the **remove_disable_clock_gating_check** command.

Multicorner-Multimode Support

This command applies to all scenarios.

EXAMPLES

The following example disables clock-gating checks on the specified cell or pin:

```
prompt> set_disable_clock_gating_check an2/Z  
prompt> set_disable_clock_gating_check or1
```

SEE ALSO

`remove_disable_clock_gating_check(2)`

set_disable_timing

Disables timing arcs in the current design.

SYNTAX

```
status set_disable_timing  
  object_list  
  [-from from_pin_name -to to_pin_name]  
  [-restore]
```

Data Types

```
object_list    collection  
from_pin_name  string  
to_pin_name    string
```

ARGUMENTS

object_list

Specifies the cells, pins, ports, library pins, or library cells through which timing is disabled. The specified cells or the cells of the specified pins are set to be size only to leave room to apply sizing on them.

-from *from_pin_name*

Specifies that only arcs from this pin on the specified cell are disabled.

This option must be used with the **-to** option.

If you use the **-from** and **-to** options, the *object_list* argument must contain only cells.

-to *to_pin_name*

Specifies that only arcs to this pin on the specified cell are disabled.

This option must be used with the **-from** option.

If you use the **-from** and **-to** options, the *object_list* argument must contain only cells.

-restore

Restores the specified arcs so that they are no longer disabled.

DESCRIPTION

The **set_disable_timing** command disables timing through the specified cells, pins, or ports in the current design.

Any cell, pin, or port in the current design or its subdesigns can be disabled. Cells at lower levels in the design hierarchy are specified as *instance1/instance2/cell_name*. Pins at lower levels in the design hierarchy are specified as *instance1/instance2/cell_name/pin_name*. Ports at lower levels in the design hierarchy are specified as *instance1/instance2/cell_name/port_name*.

Library pins and library cells can also be disabled.

Note: For subdesigns in the hierarchy, you must specify instance names instead of design names.

When the timing through a cell is disabled, only those cell arcs that lead to the output pins of the cell are disabled. When the timing through a pin or port is disabled, only those cell arcs that lead to or from that pin or port are disabled.

When the **-from** and **-to** options are specified, all arcs between these pins on the cell are disabled.

Use the **report_lib -timing_arcs** command to list the timing arcs for a library cell, and the **report_design** command to list the disabled arcs in the current design.

To enable the disabled cells, pins, or ports, use the **set_disable_timing -restore** or **reset_design** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

In the following example, all timing arcs are disabled that lead to output pins of cells U2 and U3 of the current design.

```
prompt> set_disable_timing {U2 U3}
```

In the following example, all timing arcs that lead to output pin Z of cell U5 in the current design are disabled.

```
prompt> set_disable_timing U5/Z
```

In the following example, all timing arcs between pins A and Z on cell U1/U1 in the current design are disabled.

```
prompt> set_disable_timing U1/U1 -from A -to Z
```

In the following example, all timing arcs between pins D and Q on cell FD1 in the library *my_lib* are disabled. This command disables arcs on a library cell; therefore, all instances of that library cell are affected in any design linked to the same library.

```
prompt> set_disable_timing my_lib/FD1 -from D -to Q
```

SEE ALSO

[remove_attribute\(2\)](#)

```
report_lib(2)  
reset_design(2)
```

set_domain_supply_net

Sets the primary power net and primary ground net of an already existing power domain.

SYNTAX

```
status set_domain_supply_net  
  domain_name  
  -primary_power_net supply_net_name  
  -primary_ground_net supply_net_name
```

Data Types

```
domain_name      string  
supply_net_name  string
```

ARGUMENTS

domain_name

Specify the name of the power domain.

If a power domain with the specified name does not exist, in the current scope, this command fails.

The *domain_name* option is a required option and must be specified.

-primary_power_net *supply_net_name*

Specify the power net of the power domain.

If a supply net with the specified name does not exist in the current scope, the command fails. If the supply net is not associated with specified power domain, the command fails.

This is a required option and must be specified.

-primary_ground_net *supply_net_name*

Specify the ground net of the power domain.

If a ground net with the specified name does not exist in the current scope, the command fails. If the ground net is not associated with specified power domain, the command fails.

This is a required option and must be specified.

DESCRIPTION

The `set_domain_supply_net` command enables you to set the primary power net and the primary ground net of a power domain. All extent of the power domain shares the same power/ground supply until explicitly excluded. Here "explicitly excluded" means it is explicitly connected with another supply_net (non primary power/ground supply_net) by command `connect_supply_net`. This command errors out if power domain already has a primary power and ground net.

The supply_net must be created in the same power_domain at first before it could be set as the primary power or ground supply_net. Use command `create_supply_net` to create a supply_net at a specified power_domain.

This command cannot be used to specify the supply nets of a power domain which was created using the `-supply {primary ..}` argument.

Command returns 1 on success, returns 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates two supply nets on power_domain PD1, and then sets them as a primary power or ground net.

```
prompt> create_power_domain PD1 -elements INST_1
PD1
prompt> create_supply_net A_VDD -domain PD1
A_VDD
prompt> create_supply_net A_VSS -domain PD1
A_VSS
prompt> set_domain_supply_net PD1 -primary_power_net A_VDD \
    -primary_ground_net A_VSS
1
```

SEE ALSO

`create_power_domain(2)`
`create_supply_net(2)`
`connect_supply_net(2)`

set_dont_retime

Sets the **dont_retime** attribute on cells and designs in the current design to prevent sequential cells from being moved by retiming optimizations.

SYNTAX

```
int set_dont_retime  
    object_list  
    [true | false]
```

Data Types

object_list list

ARGUMENTS

object_list

Specifies list of objects (cells or designs) on which the **dont_retime** attribute is to be set, so that the cells themselves (if they are sequential cells) or the sequential child cells of designs or hierarchical cells will not be moved by retiming optimizations, if the effective attribute value is true.

true | false

Specifies the value with which to set the **dont_retime** attribute. The default is **true**.

DESCRIPTION

This command sets the **dont_retime** attribute on cells, and designs in the current design to prevent moving of sequential cells during retiming optimizations. Setting the **dont_retime** attribute on a hierarchical cell implies "dont_retime" on all sequential cells below it that do not have **dont_retime** set to a **false** value. Setting the **dont_retime** attribute on a leaf level cell implies "dont_retime" on the cell, if it is a sequential cell. Setting the **dont_retime** attribute on a design implies "dont_retime" on all sequential cells inside that do not have **dont_retime** set to a **false** value. **dont_retime** attributes set on child cells or designs have preference over **dont_retime** attributes set on any ancestor cell or design. **dont_retime** attributes set on hierarchical cells that are instances of a design have preference over **dont_retime** attributes set on the design itself.

You can see the **dont_retime** value explicitly on cells or designs by using the **report_cell**, **report_design** or **report_attribute** command.

To remove the **dont_retime** attribute, use **remove_attribute**.

EXAMPLES

The following example specifies that the cells named *z1_reg* and *z2_reg* are not to be modified during compile.

```
prompt> set_dont_retime [get_cells {z1_reg z2_reg}] true
```

The following example specifies that the cell named *H1/z_reg* can be moved during retiming, but any other sequential cell inside cell *H1* cannot be moved.

```
prompt> set_dont_retime [get_cells "H1"] true
prompt> set_dont_retime [get_cells "H1/z_reg"] false
```

The following example specifies that no sequential cells inside any instance of design *Controller* may be moved by retiming optimizations.

```
prompt> set_dont_retime find [get_designs "Controller"] true
```

SEE ALSO

- compile(2)
- compile_ultra(2)
- report_cell(2)
- report_design(2)
- report_attribute(2)
- optimize_registers(2)
- balance_registers(2)

set_dont_touch

Sets the **dont_touch** attribute on cells, nets, references, and designs in the current design, and on library cells, to prevent modification or replacement of these objects during optimization.

SYNTAX

```
status set_dont_touch  
      object_list  
      [true | false]
```

Data Types

object_list list

ARGUMENTS

object_list

Specifies list of objects (cells, nets, references, designs, and library cells) on which the **dont_touch** attribute is to be set, so that the objects will not be modified or replaced. If more than one object name is specified, each name must be enclosed in quotation marks (" ") or braces ({}).

true | false

Specifies the value with which to set the **dont_touch** attribute. The default is **true**.

DESCRIPTION

This command sets the **dont_touch** attribute on cells, nets, references, designs in the current design, and on library cells, to prevent modification or replacement of these objects during optimization.

Setting the **dont_touch** attribute on a hierarchical cell implies "dont_touch" on all cells below it that do not have **dont_touch** set to **false**.

Setting the **dont_touch** attribute on a library cell implies "dont_touch" on all instances of that cell. Setting the **dont_touch** attribute on a net implies "dont_touch" only on mapped combinational cells connected to that net. All mapped combinational cells connected to the net must be at the same level of hierarchy.

Note that if the **enable_keep_signal_dt_net** variable is set to **true**, a different mechanism is used to preserve a dont_touch net. See the [enable_keep_signal_dt_net](#) man page for more information.

Setting the **dont_touch** attribute on a reference implies "dont_touch" on all cells of that reference during subsequent optimizations of

the design.

Setting the **dont_touch** attribute on a design has an effect only when the design is instantiated within another design as a level of hierarchy. In this case, "dont_touch" on the design implies that all cells under that level of hierarchy are "dont_touch." Setting **dont_touch** on the top-level design has no effect because it is not instantiated within any other design.

Note the following information:

- The **dont_touch** attribute is ignored on nets that have unmapped cells on them.
- The **dont_touch** attribute applied by the tool on the cells and nets in the transitive fanout of **dont_touch_network** objects overrides the **false dont_touch** attribute value that is set by the **set_dont_touch** command. You can see the **dont_touch** value on cells or nets by using the **report_cell** or **report_net** command.
- Setting the **dont_touch** attribute on unmapped cells in Design Compiler topographical mode causes the **compile_ultra** command to fail during execution.
- Setting the **dont_touch** attribute on selectops and their control logic is not supported if this logic resides at the top level of the design.
- A **dont_touch** attribute on a clock gating cell or its parent hierarchical cell will prevent clock gating optimizations from removing or modifying it.
- The **dont_touch** attribute should be removed by **remove_attribute** command. It is not equivalent to **set_dont_touch** to false which will have its meaning depending on the context. E.g. its parent has **set_dont_touch** is true. In the DC Explorer flow, **set_dont_touch** to false on a cell will be ignored if none of the cell's parent has **dont_touch** set to true. In this case, the cell may be optimized away during compile. If the cell does not get optimized away during compile, the **dont_touch** false attribute may still be lost after compile.

The **dont_touch** attribute affects DFT insertion as follows:

- The **dont_touch** attribute indirectly affects DFT insertion by preventing the cell type from being changed. It is ignored for scan stitching, test signal routing, logic insertion. It is also ignored when an identified shift register is split by the scan architect; the head scan flip-flops of any new shift register segments are scan-replaced even if they have the **dont_touchfP** attribute applied.
- A **dont_touch** attribute applied to a nonscan sequential cell keeps the cell as a nonscan cell. It will not be scan-replaced or stitched into scan chains. However, nonscan cells identified as shift register elements can be stitched into scan chains.
- A **dont_touch** attribute applied to a test-ready (scan-replaced) sequential cell keeps the cell as a scan-replaced cell. If the cell is a valid scan cell, it will be stitched into scan chains. If not (due to DRC violations or other directives such as **set_scan_element false**), it will remain as an unstitched test-ready cell.

When an object name is specified, the **set_dont_touch** command first looks within the current design for a cell that matches that name. If it finds a match, a **dont_touch** attribute is assigned to that cell. If no match is found, the command next looks within the current design for a net that matches the specified name. If a match is found, a **dont_touch** attribute is assigned to that net. If no match is found, the command next looks for a reference, then a design in the system, and finally for a library cell in the system. The tool assigns a **dont_touch** attribute to the first object for which it finds a match.

To remove the **dont_touch** attribute, use the **remove_attribute** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies that the cells named *block1* and *analog1* are not to be modified during optimization:

```
prompt> set_dont_touch [get_cells {block1 analog1}]
```

The following example specifies the removal of the **dont_touch** attribute from the cell named *block1*:

```
prompt> remove_attribute [get_cells block1] dont_touch
```

The following example specifies that the net named *N1* is not to be modified during optimization:

```
prompt> set_dont_touch [get_nets N1] true
```

SEE ALSO

- compile(2)
- get_dont_touch_cells(2)
- get_dont_touch_nets(2)
- list_dont_touch_types(2)
- report_cell(2)
- report_dont_touch(2)
- report_net(2)
- report_reference(2)
- set_dont_touch_network(2)
- remove_attribute(2)
- enable_keep_signal_dt_net(3)

set_dont_touch_network

Sets the **dont_touch_network** attribute on clocks, pins, or ports in the current design to prevent cells and nets in the transitive fanout of the **set_dont_touch_network** objects from being modified or replaced during optimization.

SYNTAX

```
status set_dont_touch_network  
  objects  
  [-no_propagate]
```

Data Types

objects collection

ARGUMENTS

objects

Specifies the clocks, pins, or ports in the current design on which to set the **dont_touch_network** attribute. If you include more than one object, they must be enclosed in either quotation marks or braces ({}).

-no_propagate

Indicates that the dont_touch network is not propagated through logic gates.

DESCRIPTION

This command sets the **dont_touch_network** attribute on clocks, pins, or ports in the current design. When a design is optimized, the tool assigns **dont_touch** attributes to all cells and nets in the transitive fanout of the **dont_touch_network** objects so that they are not modified or replaced during optimization. A **dont_touch** attribute assigned by the tool using the **set_dont_touch_network** command overrides the **false dont_touch** attribute value from the **set_dont_touch** command.

If you specify a pin or a port, all nets and cells in its transitive fanout are recursively marked **dont_touch**. This recursion propagates through combinational cells but stops at registers (without marking the registers as **dont_touch**). An element is recognized as a register only if it has setup or hold constraints. If you specify an input pin, its own cell is marked **dont_touch** but its connected net is not marked **dont_touch**. If you specify an output pin, its own cell is not marked **dont_touch** but its connected net is marked **dont_touch**.

If you specify a clock, the transitive fanout of all the clock's sources is affected. The **set_dont_touch_network** command is intended primarily for clock circuitry. Placing a **dont_touch_network** on a clock object prevents the tool from modifying the clock buffer network during optimization.

The **set_dont_touch_network** command does not prevent clock gating optimizations from modifying or removing clock gating cells. To prevent this, use the **set_dont_touch** command instead.

If you specify the **-no_propagate** option, the dont_touch network is not propagated through logic gates. These logic gates are marked as size_only.

To see the **dont_touch** value on the cells or nets, use the **report_cell** or **report_net** command. The **report_net** command displays the nets that are implicitly **dont_touch** as a result of using the **set_dont_touch_network** command. You can also use the **report_dont_touch** command to report the cells and nets with implicit **dont_touch** attributes.

Note that the **dont_touch_network** attribute does not prevent compile from mapping unmapped logic (for example, cells read in from an HDL description).

To remove the **dont_touch_network** attribute, use the **remove_attribute** command on the specific object on which the **dont_touch_network** attribute has been set.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command adds the **dont_touch_network** attribute to a port named clock_in:

```
prompt> set_dont_touch_network clock_in
```

The following command removes the **dont_touch_network** attribute from the port named clock_in:

```
prompt> remove_attribute [get_port clock_in] dont_touch_network
```

The following command adds the **dont_touch_network** attribute to a net named net1 and a cell named cell2 but not to the cell named cell1:

```
prompt> get_pins -of_objects [get_nets net1]
{cell1/Y cell2/A}
prompt> set_dont_touch_network [get_pins cell1/Y]
1
```

The following command adds the **dont_touch_network** attribute to a net named net1 and cells named cell1 and cell2 but not on the net named net0:

```
prompt> get_pins -of_objects [get_nets net1]
{cell1/Y cell2/A}
prompt> get_pins -of_objects [get_cells cell1]
{cell1/A cell1/Y}
prompt> get_nets -of_objects [get_pins cell1/A]
{net0}
prompt> set_dont_touch_network [get_pins cell1/A]
1
```

SEE ALSO

- compile(2)
- get_dont_touch_cells(2)
- get_dont_touch_nets(2)
- list_dont_touch_types(2)
- remove_attribute(2)
- report_cell(2)
- report_clock(2)
- report_design(2)
- report_dont_touch(2)
- report_net(2)
- report_port(2)
- set_dont_touch(2)

set_dont_use

Sets the **dont_use** attribute on library cells, modules, and implementations. Library objects with the **dont_use** attribute are excluded from the target library during optimization.

In addition, if you optimize a design that contains cell instances that are mapped to standard cells in the target library that have the **dont_use** attribute, those cell instances are remapped to equivalent standard cells from the target library that do not have the **dont_use** attribute.

SYNTAX

```
status set_dont_use
      [-power]
      object_list
```

Data Types

object_list list

ARGUMENTS

-power

Specifies that the specified objects should not be considered for clock-gate mapping in addition to excluding them from the target library.

By default, the objects are excluded from the target library, but are still available for clock-gate mapping.

object_list

Specifies the objects (library cells, modules, or implementations) on which to set the **dont_use** attribute.

The specified objects must be in one of the target libraries or in a library already loaded into the tool and the object names must include the library prefix. If you specify more than one object, they must be enclosed in quotes or braces ({}).

DESCRIPTION

Sets the **dont_use** attribute on the specified library objects, so that they are not used during optimization.

In addition, if you optimize a design that contains cell instances that are mapped to standard cells in the target library that have the **dont_use** attribute, those cell instances are remapped to equivalent standard cells from the target library that do not have the **dont_use** attribute.

If you use the **-power** option, the **pwr_cg_dont_use** attribute is also set on the specified objects. When this attribute is present on an object, it is not considered for clock-gate mapping.

NOTE:

This command affects only the copy of the library that is currently loaded into memory, and has no effect on the version that exists on disk. However, if the library is written out by using the **write_lib** command, any **dont_use** attributes are written out, which causes the library objects to be permanently disabled.

To remove **dont_use** attributes, use the **remove_attribute** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command disables the library cells named G1 and G2 in the tech_lib library:

```
prompt> set_dont_use {tech_lib/G1 tech_lib/G2}
```

The following command disables the implementations named imp1 and imp2 in the syn_lib/mod module:

```
prompt> set_dont_use {syn_lib/mod/imp1 syn_lib/mod/imp2}
```

The following command removes the library cells named icg_1 and icg_2 from the list of integrated clock gating cells available for clock-gate mapping.

```
prompt> set_dont_use -power {tech_lib/icg_1 tech_lib/icg_2}
```

SEE ALSO

compile(2)
remove_attribute(2)
report_lib(2)
target_library(3)

set_dp_int_round

Sets the rounding positions on the inferred multiplier and the multiplier-based instantiated DesignWare cells.

SYNTAX

```
status set_dp_int_round  
  cell_list  
  external_rounding_position  
  [internal_rounding_position]
```

Data Types

```
cell_list           object_list  
external_rounding_position  int  
internal_rounding_position  int
```

ARGUMENTS

cell_list

Sets the rounding and internal rounding position on the specified cells.

external_rounding_position

Sets the bit position for external rounding.

internal_rounding_position

Sets the bit position for internal rounding.

DESCRIPTION

This command specifies the external and internal rounding positions (IR) on the inferred multiplier and the multiplier-based instantiated DesignWare cells.

The supported cells are: MULT_UNNS_OP and MULT_TC_OP operators and the following DesignWare components: DW02_mult, DW_mult_uns, DW_mult_tc, DW_square, DW02_prod_sum, DW02_prod_sum1, DW_prod_sum_uns, DW_prod_sum_tc, DW02_mac, DW_mac_tc, DW_mac_uns, DW02_multp, DW_squarep, and DW02_tree.

Internal rounding allows a tradeoff between precision and area in arithmetic circuits. All bits lower than the internal rounding position are discarded from the internal addends. Area is saved because no logic is required to generate and add these bits. However, a rounding error is introduced, which degrades the precision of the result. An offset is automatically added to keep the

error range as symmetric and the bias as small as possible.

The result of a datapath with internal rounding is usually truncated somewhere above the internal rounding position. This truncation can also be rounded by setting the external rounding position to this bit position. Rounding is achieved by adding a constant 1 at the next lower bit position.

Internal rounding changes the functionality of a datapath block or a multiplier-like DesignWare design. Use of this command results in a design that has functional differences from the original design. Comparison of the original design with the resulting netlist will show differences in formal equivalence checking tools.

A rounding error report is printed with the **report_resources** command.

To remove the rounding attribute, use the **remove_dp_int_round** command.

A few notes about using this command:

1. The tool only supports internal rounding on generated implementations. Therefore, if you use **set_implementation** on the non-generated implementation for a instantiated DesignWare cell, the internal rounding does not happen.
2. Multipliers with internal rounding will not be extracted, merged or shared.
3. All inferred multipliers driving a common output will use the same internal rounding information. If the internal rounding values are different among those multipliers, the minimum rounding positions will be used for each multiplier. If the shared multipliers include a multiplier with a set rounding position, and other multipliers with no internal rounding information, all multipliers will use the minimum of the set rounding positions.
4. An internal rounding setting of 0 for both the external and internal rounding position has meaning. It forces that operation to use exact precision; the output of the operation will not be rounded.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the external rounding position on cell mult_6 to 7 and the internal rounding position to 5:

```
prompt> set_dp_int_round mult_6 7 5
Setting rounding position 7, internal rounding position 5 on: mult_6
1
```

The **report_resources** command reports the rounding errors.

```
prompt> report_resources
...
DesignWare rounding error report in design mult_inst_DW_mult_uns_1(cell mult_6):
Rounding Error Report for product[15:0]
```

=====								
	Relative	Relative Error			Absolute Error			
Rounding	to Bit	Min	Max	Avg	Min	Max	Avg	
=====								
Internal	5	-3.00	2.00	-0.50	-96	64	-16	

Internal	7	-0.75	0.50	-0.12	-96	64	-16	
External	7	-0.50	0.49	-0.00	-64	63	-0.5	

```
Total      7  -1.25  0.99 -0.13  -160   127  -16.5
```

```
=====
```

The behavioral Verilog file has extension .bvrl and is found in the dwsvf_* directory.

```
prompt> ls dwsvf_*
mult_inst_DW_mult_uns_1.round.bvrl
```

The following example sets the external rounding position on a multiplier mult_8. The **report_resources** command shows the original IR settings.

```
prompt> set_dp_int_round [find cell mult_8] 5
Setting rounding position 5, internal rounding position 5 on: mult_8
1
```

The **report_resources** command reports the rounding settings and errors.

```
prompt> report_resources
...
```

DW rounding error report in design test_DW_mult_uns_0(cell mult_x_8_1):
Rounding Error Report for product[11:0]

Rounding	Relative to Bit	Relative Error			Absolute Error		
		Min	Max	Avg	Min	Max	Avg
Internal	5	-2.50	1.53	-0.48	-80	49	-15.5
Internal	5	-2.50	1.53	-0.48	-80	49	-15.5
External	5	-0.50	0.47	-0.02	-16	15	-0.5
Total	5	-3.00	2.00	-0.50	-96	64	-16

SEE ALSO

```
report_resources(2)
remove_dp_int_round(2)
```

set_dp_smartgen_options

Controls the strategies used when generating the datapath cell for arithmetic and shift operators.

SYNTAX

```
status set_dp_smartgen_options
  [-all_options auto | true | false | default]
  [-booth_encoding auto | true | false]
  [-booth_radix8 auto | true | false]
  [-booth_mux_based auto | true | false]
  [-booth_cell auto | true | false]
  [-mult_radix4 auto | true | false]
  [-mult_nand_based auto | true | false]
  [-inv_out_adder_cell auto | true | false]
  [-4to2_compressor_cell auto | true | false]
  [-adder_radix auto | 2 | 3 | 4]
  [-ling_adder auto | true | false]
  [-hybrid_adder auto | true | false]
  [-carry_select_adder_cell auto | true | false]
  [-cond_sum_adder auto | true | false]
  [-sklansky_adder auto | true | false]
  [-brent_kung_adder auto | true | false]
  [-bounded_fanout_adder auto | true | false]
  [-mux_based auto | true | false]
  [-inv_adder_cell auto | true | false]
  [-sop2pos_transformation auto | true | false]
  [-tp_opt_tree auto | true | false]
  [-tp_oper_sel auto | true | false]
  [-smart_compare auto | true | false]
  [-optimize_for default | area | speed | area,speed]
  [-power_effort off | auto | medium | high]
  [-mult_arch auto | and | nand | and_radix4 | nand_radix4 | benc_radix4 | benc_radix8 | benc_radix4_mux | benc_radix8_mux]
  [-hierarchy]
  [design or cell list]
```

ARGUMENTS

-all_options auto | true | false | default

Specifies the default value for all smart generation options. Use this option with other options to provide a specific value to one option and default values to other options.

The **-all_options** argument also accepts the **default** value. This value sets all smart generation options to their defaults.

The **set_dp_smartgen_options** command always processes the value of the **-all_options** option before any other options.

-booth_encoding auto | true | false

Controls the usage of Booth-encoding architectures to implement multipliers.

When set to **auto**, the default, the tool uses the option only if there is a QoR benefit. When you set the **-booth_encoding** option to **true**, the tool always uses the option. When you set the **-booth_encoding** option to **false**, the tool never uses the option.

-booth_radix8 auto | true | false

Controls the usage of radix-8 Booth-encoding architectures to implement multipliers. This option is active only if the **-booth_encoding** option is set to **auto** or **true**.

When you set the **-booth_radix8** option to **auto**, the default, the tool uses the **-booth_radix8** option only if there is a QoR benefit. The tool always uses the **-booth_radix8** option when it is set to **true** and never uses the **-booth_radix8** option when it is set to **false**.

-booth_mux_based auto | true | false

Controls the usage of MUX-based Booth encoding. MUX-based encoding provides better QoR for technologies with fast multiplexer cells. Otherwise, the tool uses XOP-based Booth encoding.

When you set the **-booth_mux_based** option to **auto**, the default, the tool uses the option only used if it determines there is a QoR benefit. When you set the **-booth_mux_based** option to **true**, the tool always uses the option. When you set the **-booth_mux_based** option to **false**, the tool never uses the option.

-booth_cell auto | true | false

Controls the usage of special Booth multiplier cells in the target library for compilation.

When you set the **-booth_cell** option to **auto**, the default, the tool uses the option only if it determines there is a QoR benefit. When you set the **-booth_cell** option to **true**, the tool always uses the option. When you set the **-booth_cell** option to **false**, the tool never uses the option.

-mult_radix4 auto | true | false

Controls the use of radix-4 non-Booth architectures to implement multipliers. This option cannot be used if the **-booth_encoding** option is set to **true**. The **-booth_encoding** option must be inactive (set to **auto** or **false**).

When you set the **-mult_radix4** option to **auto**, the default, the tool uses the option only if it determines there is a QoR benefit. When you set the **-mult_radix4** option to **true**, the tool always uses the option. When you set the **-mult_radix4** option to **false**, the tool never uses the option. This option is only available in the DesignWare minPower flow.

-mult_nand_based auto | true | false

Controls the usage of NAND-based multiplier structures. Non-Booth multipliers use AND gates or an INVERT-NOR structure to generate the partial-product bits. This option allows you to use the NAND gates to produce inverted bits that you can add directly. This can result in lower dynamic power and lesser glitches.

When you set the **-mult_nand_based** option to **auto**, the default, the tool uses the option only when the minPower flow is enabled and if it determines there is a QoR benefit. When you set the **-mult_nand_based** option to **true**, the tool always uses the option. When you set the **-mult_nand_based** option to **false**, the tool never uses the option. The option has no effect when you use Booth encoding in multipliers.

-inv_out_adder_cell auto | true | false

Controls the usage of full-adder cells with inverted sum-and-carry outputs in the target library for compilation.

When you set the **-inv_out_adder_cell** option to **auto**, the default, the tool uses the option only if it determines there is a QoR benefit. When you set the **-inv_out_adder_cell** option to **true**, the tool always uses the option. When you set the **-**

inv_out_adder_cell option to **false**, the tool never uses the option.

-4to2_compressor_cell auto | true | false

Controls the usage of 4-2 compressor cells in the target library for compilation.

When you set the **-4to2_compressor_cell** option to **auto**, the default, the tool uses the option only if it determines there is a QoR benefit. When you set the **-4to2_compressor_cell** option to **true**, the tool always uses the option. When you set the **-4to2_compressor_cell** option to **false**, the tool never uses the option.

-adder_radix auto | 2 | 3 | 4

Controls the radix of the prefix structure in adders. Most adder architectures use a parallel-prefix structure to propagate carries. The default prefix structure has radix 2 (combines 2 bits per prefix node). Radix 3 and 4 (combine 3 or 4 bits per prefix node) result in fewer, but more complex prefix nodes. They can deliver better QoR if special cells are available in the technology to implement them.

When you set the **-adder_radix** option to **auto**, the tool automatically selects the optimal radix for best QoR. When you set the **-adder_radix** option to **2**, **3**, or **4**, the tool sets the radix accordingly. The default is **2**.

-ling_adder auto | true | false

Controls the usage of Ling adder architectures.

When you set the **-ling_adder** option to **auto**, the default, the tool uses the option only if it determines there is a QoR benefit. When you set the **-ling_adder** option to **true**, the tool always uses the option. When you set the **-ling_adder** option to **false**, the tool never uses the option.

-hybrid_adder auto | true | false

Controls the usage of hybrid parallel-prefix or carry-select adder architectures (spanning-tree adders).

When you set the **-hybrid_adder** option to **auto**, the default, the tool uses the option only if it determines there is a QoR benefit. When you set the **-hybrid_adder** option to **true**, the tool always uses the option. When you set the **-hybrid_adder** option to **false**, the tool never uses the option.

-carry_select_adder_cell auto | true | false

Controls the usage of carry-select adder cells in the target library for compilation.

When you set the **-carry_select_adder_cell** option to **auto**, the default, the tool uses the option only if it determines there is a QoR benefit. When you set the **-carry_select_adder_cell** option to **true**, the tool always uses the option. When you set the **-carry_select_adder_cell** option to **false**, the tool never uses the option.

-cond_sum_adder auto | true | false

Controls the usage of conditional-sum adder architectures.

When you set the **-cond_sum_adder** option to **auto**, the default, the tool uses the option only if it determines there is a QoR benefit. When you set the **-cond_sum_adder** option to **true**, the tool always uses the option. When you set the **-cond_sum_adder** option to **false**, the tool never uses the option.

-sklansky_adder auto | true | false

Controls the usage of Sklansky parallel-prefix adder architectures. This is a static architecture that is not optimized for context. It usually results in QoR degradation, but it can be more predictable.

When you set the **-sklansky_adder** option to **auto**, the tool uses the option only if it determines there is a QoR benefit. When you set the **-sklansky_adder** option to **true**, the tool always uses the option. When you set the **-sklansky_adder** option to **false**, the default, the tool never uses the option.

-brent_kung_adder auto | true | false

Controls the usage of Brent-Kung parallel-prefix adder architectures. This is a static architecture that is not optimized for context. It usually degrades QoR, though it is more predictable.

When you set the **-brent_kung_adder** option to **auto**, the tool uses the option only if it determines there is a QoR benefit. When you set the **-brent_kung_adder** option to **true**, the tool always uses the option. When you set the **-brent_kung_adder** option to **false**, the default, the tool never uses the option.

-bounded_fanout_adder auto | true | false

Controls the usage of bounded fanout adder architectures (Kogge-Stone parallel-prefix architectures).

When you set the **-bounded_fanout_adder** option to **auto**, the tool uses the option only if it determines there is a QoR benefit. When you set the **-bounded_fanout_adder** option to **true**, the tool always uses the option. When you set the **-bounded_fanout_adder** option to **false**, the default, the tool never uses the option.

-mux_based auto | true | false

Controls the usage of MUX-based architectures by datapath generators.

When you set the **-mux_based** option to **auto**, the default, the tool uses the option only if it determines there is a QoR benefit. When you set the **-mux_based** option to **true**, the tool always uses the option. When you set the **-mux_based** option to **false**, the tool never uses the option.

-inv_adder_cell auto | true | false

Controls the usage of full adder cells with inverted carry-in or carry-out in the target library for compilation.

When you set the **-inv_adder_cell** option to **auto**, the default, the tool uses the option only if it determines there is a QoR benefit. When you set the **-inv_adder_cell** option to **true**, the tool always uses the option. When you set the **-inv_adder_cell** option to **false**, the tool never uses the option.

-sop2pos_transformation auto | true | false

Controls the usage of "sop to pos" optimization strategies in datapath blocks. An example of a "sop 2 pos" transformation is as follows:

$$a*b + a*c \rightarrow (b + c) * a$$

This optimization can improve design area, at the cost of delay.

When you set the **-sop2pos_transformation** option to **auto**, the default, the tool uses the option only if it determines there is a QoR benefit. When you set the **-sop2pos_transformation** option to **true**, the tool always uses the option. When you set the **-sop2pos_transformation** option to **false**, the tool never uses the option.

-tp_opt_tree auto | true | false

Controls the usage of carry-save adder tree for low power optimization based on transition probabilities (TP). This optimization reduces internal switching activity and therefore the dynamic power in SOPs when the inputs have a non-uniform transition probability distribution.

When you set the **-tp_opt_tree** option to **auto**, the default, the tool uses the option only if it determines a QoR benefit and optimized dynamic power. When you set the **-tp_opt_tree** option to **true**, the tool always uses the option if dynamic power is optimized. When you set the **-tp_opt_tree** option to **false**, the tool never uses the option.

-tp_oper_sel auto | true | false

Controls the selection or ordering of multiplier input operands for low power based on transition probability (TP). This optimization can reduce internal switching activity and therefore dynamic power in booth multipliers when the inputs have significantly different transition probabilities.

When you set the **-tp_oper_sel** option to **auto**, the default, the tool uses the option only if it determines a QoR benefit and

optimized dynamic power. When you set the **-tp_oper_sel** option to **true**, the tool always uses the option if dynamic power is optimized. When you set the **-tp_oper_sel** option to **false**, the tool never uses the option. The option only affects multipliers with booth encoding.

-smart_compare auto | true | false

Controls the ordering of comparator inputs for better timing. Depending on the logic depth of the comparator logic, an inverter is required at the output. Flipping the inputs eliminates the need for the inverter and shortens the critical path.

When you set the **-smart_compare** option to **auto**, the default, or **true**, the tool uses the option only if it determines a QoR benefit. When you set the **-smart_compare** option to **false**, the tool never uses the option.

-optimize_for default | area | speed | area,speed

Supplies a specific optimization goal for the specified cells. This option overrides the default optimization goals for the specified cell or cells.

-power_effort off | auto | medium | high

Supplies a specific power optimization goal for the specified cells. This option takes effect in the DesignWare minPower flow. It guides the generator on power optimization. When the effort is **off**, the generator does not optimize the netlist for power. When the effort is **medium**, the generator sacrifices delay for power optimization. When the effort is **high**, the generator sacrifices more delay for power.

When you set the power effort to **medium** or **high**, the tool calculates the delay or power tradeoff on the generated netlist. After the generated design is ungrouped, Design Compiler might recover the timing and give back the power gain. Therefore, the power savings from **medium** and **high** effort is not guaranteed. The delay or power tradeoff feature is most useful for designs that can meet the timing constraint. Note: DC Explorer does not support this option.

-mult_arch auto | and | nand | and_radix4 | nand_radix4 | benc_radix4 | benc_radix8 | benc_radix4_mux | benc_radix8_mux

Specifies the multiplier architecture to be implemented. This option is a concise representation for a subset of six smart generation options.

The different **-mult_arch** options translates to the following combinations of smart generation settings: **and**: AND-based non-booth architecture -booth_encoding false -mult_radix4 false -mult_nand_based false **nand**: NAND-based non-booth architecture -booth_encoding false -mult_radix4 false -mult_nand_based true **and_radix4**: AND-based radix-4 non-booth architecture -booth_encoding false -mult_radix4 true -mult_nand_based false **nand_radix4**: NAND-based radix-4 non-booth architecture -booth_encoding false -mult_radix4 true -mult_nand_based true **benc_radix4**: radix-4 booth architecture -booth_encoding true -booth_radix8 false -booth_mux_based false **benc_radix8**: radix-8 booth architecture -booth_encoding true -booth_radix8 true -booth_mux_based false **benc_radix4_mux**: MUX-based radix-4 booth architecture -booth_encoding true -booth_radix8 true -booth_mux_based true **benc_radix8_mux**: MUX-based radix-8 booth architecture -booth_encoding true -booth_radix8 true -booth_mux_based true

When combined with other smart generation options related to multiplier architecture, the **-mult_arch** option takes precedence. For example, the following command sets the **-booth_encoding** option to **true**: `prompt> set_dp_smartgen_options -mult_arch benc_radix8 -booth_encoding false`

Note: The **-mult_radix4** option, when used either individually or with the **-mult_arch** option, is available only when you enable the minPower flow. If the minPower flow is not enabled, the tool drops the **-mult_radix4** option when you run the **compile_ultra** command.

-hierarchy

By default, when the smart generation options are specified on a design or a hierarchical cell, the specified options are only applied to the current level of hierarchy of the specified objects. When this option is specified, the command automatically applies the smart generation options on all the child instances of the specified objects.

design or cell list

An optional list of cells and designs. When present, the switch settings in this command affect only the specified objects.

If you set conflict smart generation options on a design and an instance that instantiates this design, you get a warning message. The setting on the instance is used by the generator.

DESCRIPTION

The **set_dp_smartgen_options** command controls smart generation strategies used in datapath synthesis.

The default value for all smart generation strategies is **auto**. By default, the best strategies are automatically chosen based on the given design constraints.

When the smart generation options are specified without an object list, the specified options are automatically applied to the entire design through all level of hierarchy regardless of the use of **-hierarchy** option. When the smart generation options are specified with an object list, the specified options are applied only on the specified objects. The object can be a design, a hierarchical cell, or a synthetic operator. The options do not take effect when they are applied on the generic GTECH cell or the cell that is already mapped to the technology library.

When the options are applied on a synthetic operator, most options take effect only when the operator is not extracted into a DP_OP cell. When the operator is extracted into DP_OP cell, only the **-optimize_for** option and the options that control multiplier architecture (**-mult_arch**, **-booth_encoding**, **-booth_radix8**, **-booth_mux_based**, **-booth_cell**, **-mult_radix4**, and **-mult_nand_based**) will be inherited to the DP_OP cell. When the options are copied to the DP_OP cell, the options are applied on all operators within the same DP_OP cell.

When synthetic operators with different option values are extracted into a DP_OP cell, the DP_OP cell takes the option value based on the following priority: For the **-optimize_for** option, the option value **speed** has the highest priority, **area,speed** is the next, and **area** has the lowest priority. For the options that control multiplier architectures, the option value **true** has the highest priority. **false** is the next, and **auto** has the lowest priority.

The synthetic operators could go through multiple complex transformations during **compile_ultra**. For example, two operators may be shared at the beginning and unshared later depending on the design constraints. When the synthetic operator with the smart generation options go through such transformations, it is possible that the initial option setting is not preserved. Pay attention to the **SYNOPT** messages in the log file. The message is generated when the smart generation options on the synthetic operators are changed during **compile_ultra**.

When you specify the multiple smart generation options on the same object, all options must be set by a single command, except the **-optimize_for** and **-power_effort** options. Only the **-optimize_for** and **-power_effort** options can be set using a separate **set_dp_smartgen_options** command.

When the smart generation options are specified on the object that already has other smart generation options, the options set by the previous command are removed, and only the options set by the new command are applied. The following example removes the **-mult_arch** option from the U1 cell and apply only the **-4to2_compressor** option: `prompt> set_dp_smartgen_options -mult_arch benc_radix4 [get_cells U1]` `prompt> set_dp_smartgen_options -4to2_compressor_cell true [get_cells U1]` Warning: Overriding the settings from the previous **set_dp_smartgen_options** command on U1. (UISN-84)

All options are applied on the U1 cell in the following example: `prompt> set_dp_smartgen_options -mult_arch benc_radix4 -4to2_compressor_cell true [get_cells U1]` `prompt> set_dp_smartgen_options -optimize_for speed [get_cells U1]` `prompt> set_dp_smartgen_options -power_effort high [get_cells U1]`

The option values set by **set_dp_smartgen_options** command are listed in the datapath optimization report by **report_resources** command.

EXAMPLES

The following example uses **set_dp_smartgen_options** to control the usage of booth encoded architectures when generating multipliers.

```
prompt> set_dp_smartgen_options -booth_encoding true
```

The following example sets all the options to **auto**, except for **-booth_encoding**, which is set to **true** and overrides the value of **-all_options**:

```
set_dp_smartgen_options -all_options auto -booth_encoding true
```

SEE ALSO

report_dp_smartgen_options(2)

set_drive

Sets the **rise_drive** or **fall_drive** attributes to the specified resistance values on the specified input and inout ports.

SYNTAX

```
status set_drive  
    resistance  
    [-rise] [-fall] [-min] [-max]  
    port_list
```

Data Types

```
resistance    float  
port_list     list
```

ARGUMENTS

resistance

Specifies a nonnegative resistance value to which the **rise_drive** or **fall_drive** attributes are to be set on the ports in *port_list*. The *resistance* is the output resistance of the cell that drives the port, such that a higher drive strength (resistance) means less drive capability and longer delays. Thus, a *resistance* of 0 is infinite drive, or no delay between the ports and all that is connected to them. The *resistance* must be in unit consistent with the technology library used during optimization.

-rise -fall

Indicates that the **rise_drive** or **fall_drive** attributes of *port_list* are to be set to *resistance*. If you don't specify either, both are set to *resistance*.

-min -max

Indicates that the drive strength is to be applied for minimum or maximum delay analysis. If no value is specified for minimum analysis, the maximum value is used.

port_list

Specifies a list of input or inout port names of the current design on which the drive attributes are to be set. If you specify more than one port, you must enclose the ports in either quotes or braces ({}).

DESCRIPTION

Sets the **rise_drive** or **fall_drive** attributes to *resistance* on specified input and inout ports in the current design.

Note: The **set_driving_cell** command is more convenient and accurate than **set_drive** for describing the drive capability of a port. The **set_drive** command removes any corresponding rise or fall driving cell attributes on the specified ports. Use **set_driving_cell** instead of **set_drive**, if possible.

During optimization, the drive of an input port is used to calculate the timing delay to gates driven by that port. The delay to these gates is computed as follows for the generic CMOS delay model: $\text{Time} = \text{arrival\_time} + [\text{drive} * \text{load}(\text{net})] + \text{connect\_delay}$ (if any)

To view drive information on ports, use **report_port -drive**. To remove drive attributes from ports, use **remove_attribute**. The **reset_design** command removes all attributes from a design, including the drive attributes.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example sets the rise and fall drives of ports A, B, and C to 2.0.

```
prompt> set_drive 2.0 {A B C}
```

The following example sets the rise and fall drives of all input ports to 2 and sets the rise drive on port B to 1.

```
prompt> set_drive 2 [all_inputs]
prompt> set_drive -rise 1 [get_ports B]
```

The following example sets both the rise and fall drives of port C to the corresponding drives of pin OUT of the INVERTER cell from the TECH_LIBRARY technology library using the **drive_of** command.

```
prompt> set_drive -rise drive_of(-rise TECH_LIBRARY/INVERTER/OUT) \
[get_ports C]
prompt> set_drive -fall drive_of(-fall TECH_LIBRARY/INVERTER/OUT) \
[get_ports C]
```

SEE ALSO

- all_inputs(2)
- drive_of(2)
- remove_attribute(2)
- report_port(2)
- reset_design(2)
- set_driving_cell(2)
- set_load(2)

set_driving_cell

Sets attributes on input or inout ports of the current design that specify that a library cell or output pin of a library cell drives the specified ports.

SYNTAX

```
status set_driving_cell
  [-lib_cell lib_cell_name]
  [-library lib]
  [-rise]
  [-fall]
  [-min]
  [-max]
  [-pin pin_name]
  [-from_pin from_pin_name]
  [-dont_scale]
  [-no_design_rule]
  [-none]
  [-input_transition_rise rtran]
  [-input_transition_fall ftran]
  [-multiply_by factor]
  port_list
  [-cell obsolete_-please_use_-lib_cell_instead]
```

Data Types

```
lib_cell_name  string
lib            string
pin_name       string
from_pin_name string
rtran          float
ftran          float
factor         float
port_list      list
```

ARGUMENTS

-lib_cell *lib_cell_name*

Specifies the name of the library cell to use to drive ports. You can use this option with the **-pin** option when the cell has more than one output pin to set the **driving_cell_rise** and the **driving_cell_fall** string attributes to *lib_cell_name* on the ports. To specify different cells for the rising and falling cases, execute the command twice, once using the **-rise** option and once using the **-fall** option, specifying the appropriate *lib_cell_name* for each. When you use this option, you must also use the **-dont_scale** and **-multiply_by** options. By default, the command searches the libraries in **link_library** for the cell.

-library *lib*

Specifies the library name or a collection of libraries in which to find the name of the library cell to use to drive ports (*lib_cell_name*). If the lib cell can be found in multiple libraries, the library with matching operating condition on the port will be used. When no library is specified, all the libraries will be searched for the lib cell with matching operating condition. If a library cell with a matching operating condition cannot be found, the first one found with a matching lib cell name will be used.

You can use this option only with the **-lib_cell** option. It sets the **driving_cell_library_rise** and the **driving_cell_library_fall** string attributes to the library name on the ports. To specify different libraries for the rising and falling cases, execute the command twice, using the **-rise** option once and the **-fall** option once, specifying the appropriate *lib* for each.

-rise

Specifies that the *lib_cell_name*, *lib*, *pin_name*, and *from_pin_name* correspond to the rising case, and sets the **driving_cell_rise**, **driving_cell_library_rise**, **driving_cell_pin_rise**, and **driving_cell_from_pin_rise** attribute strings, respectively, on the objects. You can use this option with the **-fall** option to specify both the rising and falling cases.

-fall

Specifies that the *lib_cell_name*, *lib*, *pin_name*, and *from_pin_name* correspond to the falling case, and sets the **driving_cell_fall**, **driving_cell_library_fall**, **driving_cell_pin_fall**, and **driving_cell_from_pin_fall** attribute strings, respectively, on the objects. You can use this option with **-rise** to specify both the rising and falling cases.

-min

Sets driving cell information for analysis at the minimum operating condition only.

-max

Sets driving cell information for analysis at the maximum operating condition only.

-pin *pin_name*

Specifies the output pin on the driving cell that is to drive the ports. The **-pin** option is required when you use the **-lib_cell** option and the driving cell has more than one output pin or when using the **-from_pin** option. This option sets the **driving_cell_pin_rise** and the **driving_cell_pin_fall** string attributes to *pin_name* on the ports. To specify different pins for the rising and falling cases, execute the command twice, once using the **-rise** option and once using the **-fall** option, specifying the appropriate *pin_name* for each. The default is to use the first timing arc found on the library cell.

-from_pin *from_pin_name*

Specifies the input pin on the driving cell to use when finding a timing arc. This option is required when the driving cell has more than one input pin and the arcs from those pins have different drive characteristics, and when using the **-pin** and **-lib_cell** options. The **-from_pin** option sets the **driving_cell_from_pin_rise** and **driving_cell_from_pin_fall** string attributes to *from_pin_name* on the ports. The default is to use the first timing arc found on the library cell.

-dont_scale

Specifies that the timing analyzer is not to scale the drive capability of the ports according to the current operating conditions. You can use this option only with the **-lib_cell** option. The **-dont_scale** option sets the **driving_cell_dont_scale** Boolean attribute to **true** on the ports. By default, the port drive capability is scaled for operating conditions exactly as the driving cell would be scaled.

-no_design_rule

Indicates that the design rules associated with the driving cell are not to be applied to the driven port. This option sets the **driving_cell_no_drc** Boolean attribute to **true** on the ports. Timing-related attributes are still applied. The tool issues a warning if this option is not used. By default, design rule attributes (**max_fanout**, **max_capacitance**, **max_transition**, **min_fanout**, **min_capacitance**, **min_transition**) are derived from the driving cell and its library and applied to the port.

-none

Removes previous driving cell information.

-input_transition_rise *rtran*

Specifies the input rise transition time associated with the **-from_pin** option. Use the **-input_transition_rise** and **-input_transition_fall** options to obtain a more accurate transition time and delay time at the output pin by capturing the accurate transition time associated with the **-from_pin**. The default value is **0**.

-input_transition_fall *ftran*

Specifies the input fall transition time associated with the **-from_pin** option. The default value is **0**.

-multiply_by *factor*

Specifies a factor by which to multiply the delay characteristics of the ports. You can use this option only with the **-lib_cell** option. It affects both the load delay and the transition times of the port. It sets the **driving_cell_multiply_by** float attribute to the specified factor on the ports. The default is **1.0**.

port_list

Specifies a list of names of input or inout ports in the current design on which the driving cell attributes are to be placed. If more than one port is specified, they must be enclosed in either quotes or braces ({}).

DESCRIPTION

This command sets attributes on the specified input or inout ports in the current design to associate an external driving cell with the ports. The drive capability of the port is the same as if the specified driving cell were connected in the same context to allow accurate modeling of the port drive capability for nonlinear delay models. For the CMOS2 delay model, the edge rate of pins driven by the port is the same as if the driving cell were substituted for the port. Unless you specify the **-dont_scale** option, the drive capability of the port is scaled according to the current operating conditions.

To view drive information on ports, use the **report_port** command with the **-drive** option.

You can use the **characterize** command to automatically set driving cell attributes on subdesign ports based on their context in the entire design.

You can use the **remove_driving_cell** or **reset_design** command to remove driving cell attributes on ports. The **remove_driving_cell** command removes any corresponding rise or fall drive resistance attributes (from the **set_drive** command) on the specified ports. It is considered best practice to use the **set_driving_cell** command instead of **set_drive**, because **set_driving_cell** is more accurate.

In UPF mode, driving cell attribute is applied on the input ports only if the port's supply voltage (supply is specified using **set_related_supply_net** or **set_port_attributes -driver_supply** and the voltage is specified using **set_voltage** command) matches with the voltage of the driving cell. It will be an error if the voltages do not match and the command fails on ports not meeting this voltage compliance check.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example associates the drive capability of the AND2 library cell with the IN1 port.

```
prompt> set_driving_cell -lib_cell AND2 {IN1}
```

The following example associates the drive capability of the INV/Z library pin from the tech_lib library with all input and inout ports.

```
prompt> set_driving_cell -lib_cell INV -pin Z \  
-library tech_lib [all_inputs]
```

The following example associates the drive capability of the INV library cell with the IN1 port and specifies that the delay values should not be scaled by operating condition factors.

```
prompt> set_driving_cell -lib_cell INV -dont_scale {IN1}
```

The following example associates the Z pin of the BUF1_TS library cell with the IN1 port for rising delays and the Q pin of the DFF_TS library cell for falling delays.

```
prompt> set_driving_cell -rise -lib_cell BUF1_TS -pin Z  
prompt> set_driving_cell -fall -lib_cell DFF_TS -pin Q {IN1}
```

The following example associates the AN2 driving cell with the top1 input port with an input rise transition time of 2.3 on pin A.

```
prompt> set_driving_cell -lib_cell AN2 \  
-input_transition_rise 2.3 -from_pin A top1
```

SEE ALSO

- all_inputs(2)
- characterize(2)
- remove_driving_cell(2)
- report_port(2)
- reset_design(2)
- set_drive(2)
- set_load(2)
- set_max_capacitance(2)
- set_max_fanout(2)
- set_max_transition(2)
- set_min_capacitance(2)
- set_port_fanout_number(2)

set_dynamic_optimization

Enables or disables dynamic-power optimization on the current design.

SYNTAX

status **set_dynamic_optimization**
true | false

ARGUMENTS

true | false

Controls whether dynamic-power optimization is enabled.

When the setting is **true**, enables dynamic-power optimization. When the setting is **false**, disables dynamic-power optimization.

DESCRIPTION

This command enables or disables dynamic-power optimization on the current design. This command does not support scenario-specific settings in a multicorner-multimode design. To perform scenario-dependent dynamic-power optimization, use the **set_scenario_option -dynamic_power true** command.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example enables dynamic-power optimization in a non-multicorner-multimode design.

```
prompt> set_dynamic_optimization true
1
```

The following example disables dynamic-power optimization in a non-multicorner-multimode design.

```
prompt> set_dynamic_optimization false
1
```

SEE ALSO

set_leakage_optimization(2)
set_scenario_options(2)

set_equal

Defines two input ports as logically equivalent.

SYNTAX

```
int set_equal  
    port1  
    port2
```

Data Types

```
port1  string  
port2  string
```

ARGUMENTS

port1 port1

Two logically-equivalent input port in the current design.

DESCRIPTION

Defines two input ports in the current design as logically equivalent. This information is used to eliminate redundant ports, improving optimization quality.

Logical inconsistencies are checked for in relationships between ports. Specifying an inconsistent logical relationship between ports is not allowed.

Use the **reset_design** command to remove this property from a port.

EXAMPLES

The following example sets two input ports "A" and "B" logically equal:

```
prompt> set_equal A B
```

The following example sets up a contradictory relationship between two ports and creates an error:

```
prompt> set_equal A B  
prompt> set_opposite A B  
Warning: Can't set equal ports opposite in design 'example.ddc': 'A' 'B'
```

SEE ALSO

reset_design(2)
set_opposite(2)

set_equivalent

Declares that supply nets or supply sets are electrically or functionally equivalent.

SYNTAX

```
string set_equivalent  
  [-nets supply_net_name_list]  
  [-sets supply_set_name_list]  
  [-function_only]
```

Data Types

```
supply_net_name_list  collection  
supply_set_name_list collection
```

ARGUMENTS

-nets *supply_net_name_list*

Specifies a list of supply ports, supply nets, supply set functions, and macro PG pin names that are equivalent.

-sets *supply_set_name_list*

Specifies a list of supply set names that are equivalent.

-function_only

Specifies that the supply set is only functionally equivalent.

DESCRIPTION

The **set_equivalent** command declares that two or more supplies are equivalent. This means that they always have the same state at any given time.

If the **-nets** is specified, the command defines equivalence for a list of supply ports and supply nets. If the **-sets** is specified, the command defines equivalence for a list of supply sets and supply set handles. One or the other of these options, but not both, can be specified.

If the **-function_only** option is specified, the supplies are only declared to be functionally equivalent; otherwise, the supplies are declared to be electrically equivalent, which implies that they are also functionally equivalent.

It is an error to declare that supplies are equivalent if other constraints contradict this equivalence.

EXAMPLES

```
prompt> set_equivalent -nets {VDD1 VDD2}
```

```
prompt>
```

```
prompt> set_equivalent -sets {sstop ssex} -function_only
```

```
prompt>
```

SEE ALSO

[create_supply_net\(2\)](#)

[create_supply_set\(2\)](#)

[set_isolation\(2\)](#)

set_extraction_options

Sets the parameters that influence parasitic data extraction.

SYNTAX

```
status set_extraction_options
  [-max_process_scale max_process_scaling]
  [-min_process_scale min_process_scaling]
  [-reference_direction vertical | horizontal | use_from_tluplus]
  [-default]
```

Data Types

```
max_process_scaling  float
min_process_scaling  float
```

ARGUMENTS

-max_process_scale *max_process_scaling*

Specifies the maximum process scaling factor. The default is 1.0.

-min_process_scale *min_process_scaling*

Specifies the minimum process scaling factor. The default is 1.0.

-reference_direction vertical | horizontal | use_from_tluplus

Specifies the reference direction. Based on this information, the tool applies orientation-based etch tables. The reference direction applies to all scenarios in the multicorner-multimode flow. The default is **use_from_tluplus**.

vertical - Specifies that the reference direction is vertical when applying the orientation-based etch tables, regardless of the **REFERENCE_DIRECTION** setting in the TLUPlus files.

horizontal - Specifies that the reference direction is horizontal when applying the orientation-based etch tables, regardless of the **REFERENCE_DIRECTION** setting in the TLUPlus files.

use_from_tluplus - Specifies that the reference direction is defined by the **REFERENCE_DIRECTION** setting in the TLUPlus files. The TLUPlus files are specified by the **set_tlu_plus_files** command.

-default

Resets all options to their defaults.

DESCRIPTION

The **set_extraction_options** command specifies parameters that influence parasitic capacitance and resistance extraction in the Design Compiler topographical tool.

You can specify process scaling factors with the **-max_process_scale** and **-min_process_scale** options. The lengths and widths of the interconnect wires are scaled by the specified factors, without affecting interconnect wire depths or metal geometries inside standard cells and macros. You can use process scaling factors to adjust RC estimation for a process shrink, such as shrinking from 90 nm to 65 nm.

Use this command with the **set_tlu_plus_files** command.

You can report the settings made with the **set_extraction_options** command by using the **report_extraction_options** command.

Limitations

The command is available only in DC-Topographical mode.

Multicorner-Multimode Support

This command applies to both active and inactive scenarios.

EXAMPLES

The following example runs the **set_extraction_options** command with a **-max_process_scale** factor of 0.8 and a **-min_process_scale** factor of 0.7:

```
prompt> set_extraction_options -max_process_scale 0.8 -min_process_scale 0.7
```

SEE ALSO

extract_rc(2)
read_parasitics(2)
report_extraction_options(2)
set_delay_estimation_options(2)
set_tlu_plus_files(2)

set_false_path

Removes timing constraints from particular paths.

SYNTAX

```
status set_false_path
  [-rise | -fall] [-setup | -hold]
  [-from from_list
   | -rise_from rise_from_list
   | -fall_from fall_from_list]
  [-through through_list]
  [-rise_through rise_through_list]
  [-fall_through fall_through_list]
  [-to to_list
   | -rise_to rise_to_list
   | -fall_to fall_to_list]
  [-reset_path]
  [-comment comment_string]
```

Data Types

```
from_list      list
rise_from_list list
fall_from_list list
through_list   list
rise_through_list list
fall_through_list list
to_list        list
rise_to_list   list
fall_to_list   list
comment_string string
```

ARGUMENTS

-rise

Marks rising delays false, as measured on the path endpoint. If you do not specify either **-rise** or **-fall**, both rise and fall timing are marked false.

-fall

Marks falling delays false, as measured on the path endpoint. If you do not specify either **-rise** or **-fall**, both rise and fall timing are marked false.

-setup

Marks setup (maximum) paths false. **-setup** disables setup checking for specified paths. If you do not specify either **-setup** or **-hold**, both setup and hold timing are marked false.

-hold

Marks hold (minimum) paths false. **-hold** disables hold checking for specified paths. If you do not specify either **-setup** or **-hold**, both setup and hold timing are marked false.

-from *from_list*

Specifies start points (clocks, ports, pins, or cells) of disabled paths. If you do not specify a *from_list*, all paths to end points in *to_list* are disabled. *from_list* can include clocks, pins, or ports. If you specify a clock, all path startpoints related to the specified clock are affected. If you specify an internal pin, the pin must be a path startpoint (the clock pin of a flip-flop, for example). If a cell is specified, one path startpoint on that cell is affected.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through *through_list*

A list of path throughpoints (port, pin, or leaf cell names) of the current design. The false path applies only to paths that pass through one of the points in the *through_list*. If more than one object is included, you must enclose the objects in quotes or in '{}' braces. If you specify the **-through** option multiple times, the false path setting applies to paths that pass through a member of each *through_list* in the order the lists were given. That is, the path must first pass through a member of the first *through_list*, then through a member of the second list, and so on for every through list specified. If you use the **-through** option in combination with the **-from** or **-to** options, the false path applies only if the **-from** or **-to** conditions are satisfied and the **-through** conditions are satisfied.

-rise_through *rise_through_list*

Same as the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than once in a single command invocation as with the **-through** option.

-fall_through *fall_through_list*

Same as the **-through** option, but applies only to paths with a falling transition at the specified objects. You can specify **-fall_through** more than once in a single command invocation as with the **-through** option.

-to *to_list*

Specifies end points (clocks, ports, pins, or cells) of paths disabled. If you do not specify *to_list*, all paths from start points in *from_list* are disabled. The *to_list* can include clocks, pins, or ports. If you specify a clock, all path endpoints related to the specified clock are considered. If you specify an internal pin, the pin must be a path endpoint (for example, the data pin of a flip-flop). If you specify a cell, one path endpoint on that cell is affected.

-rise_to *rise_to_list*

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-fall_to *fall_to_list*

Same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-reset_path

Removes existing point-to-point exception information on the specified paths. Only information of the same rise/fall or setup/hold type is reset. This is equivalent to using the **reset_path** command with similar arguments before the **set_false_path** is issued.

-comment *comment_string*

Allows the command to accept a comment string. The tool honors the annotation and preserves it with the SDC object so that the exact string is written out when the constraint is written out when you use the **write_sdc** or **write_script** command. The comment remains intact through the synthesis, place-and-route, and timing-analysis flows.

DESCRIPTION

Removes timing constraints from specified paths that you know do not affect circuit operation. **set_false_path** can disable both maximum delay (setup) checking and minimum delay (hold) checking.

The **set_false_path** command disables timing from path startpoints through path throughpoints to path endpoints. Path startpoints are input ports or register clock pins. Path throughpoints can be cells, pins, or ports. Path endpoints are register data pins or output ports.

To disable the timing at a particular cell in the design, use **set_disable_timing**. This removes certain timing arcs on a cell from the timing graph, so that paths along those arcs are not traced. The **set_false_path** command still allows tracing of the paths, but removes any timing constraints on them.

set_false_path is a point-to-point timing exception command. This means it assists in overriding the default single-cycle timing relationship for one or more timing paths. Other point-to-point timing exception commands include **set_max_delay**, **set_min_delay**, and **set_multicycle_path**.

If a path satisfies multiple timing exceptions, the following rules assist in determining which exceptions take effect. Rules referring to **-from** apply equally to **-rise_from** and **-fall_from**, and similarly for the rise and fall options of **-through** and **-to**.

1. Two **group_path** commands might conflict with each other. But a **group_path** exception by itself does not conflict with another type of exception. So the remaining rules apply for two **group_path** exceptions, or two non-**group_path** exceptions.
2. If both exceptions are **set_false_paths**, there is no conflict.
3. If one exception is a **set_max_delay** and the other is **set_min_delay**, there is no conflict.
4. If one exception is a **set_multicycle_path -hold** and the other is **set_multicycle_path -setup**, there is no conflict.
5. If one exception is a **set_false_path** and the other is not, the **set_false_path** takes precedence.
6. If one exception is a **set_max_delay** and the other is not, the **set_max_delay** takes precedence.
7. If one exception is a **set_min_delay** and the other is not, the **set_min_delay** takes precedence.
8. If one exception has a **-from pin** or **-from cell** and the other does not, the former takes precedence.
9. If one exception has a **-to pin** or **-to cell** and the other does not, the former takes precedence.
10. If one exception has any **-through** points and the other does not, the former takes precedence.
11. If one exception has a **-from clock** and the other does not, the former takes precedence.

12. If one exception has a **-to clock** and the other does not, the former takes precedence.

13. The exception with the more restrictive constraint then takes precedence. For **set_max_delay** and **set_multicycle_path -setup**, this is the constraint with the lower value. For **set_min_delay** and **set_multicycle_path -hold**, it is the constraint with the higher value.

To undo the effect of **set_false_path**, use **reset_path** or **reset_design**.

Use **report_timing_requirements** to list the point-to-point exceptions on a design.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes timing constraints on paths from cell ff12 to cell ff34.

```
prompt> set_false_path -from {ff12} -to {ff34}
```

The following example removes timing constraints on paths through pin u14/Z to pin ff29/Reset that are rising at the endpoint.

```
prompt> set_false_path -rise -through {u14/Z} -to {ff29/Reset}
```

The following example disables hold checking (minimum delay timing) for endpoints clocked by PHI1. The flip-flops and latches clocked by PHI1 are checked for setup violations, but not for hold violations.

```
prompt> set_false_path -hold -to [get_clocks PHI1]
```

The following example removes timing constraints for all paths that first pass through either pin u1/Z or u2/Z and then pass through pin u5/Z or u6/Z.

```
prompt> set_false_path -through {u1/Z u2/Z} -through {u5/Z u6/Z}
```

The following example disables rising timing paths through pin U14/Z to pin ff29/Reset.

```
prompt> set_false_path -rise -through {U14/Z} -to {ff29/Reset}
```

SEE ALSO

reset_design(2)
reset_path(2)
set_disable_timing(2)
set_max_delay(2)
set_min_delay(2)
set_multicycle_path(2)

set_fanout_load

Sets the **fanout_load** attribute on the specified output ports of the current design.

SYNTAX

```
status set_fanout_load
      value
      ports
```

Data Types

<i>value</i>	float
<i>ports</i>	collection

ARGUMENTS

value

Specifies the value for the **fanout_load** attribute. The *value* argument must be expressed in units consistent with the **max_fanout** and **fanout_load** values in the technology library used during optimization.

ports

Specifies the ports in the current design on which to set the **fanout_load** attribute.

DESCRIPTION

Sets the **fanout_load** attribute to the specified value on the specified output ports in the current design. The optimization process tries to ensure that the sum of this value together with all **fanout_load** attributes for pins connected to the driver that drives this port does not exceed the driving pin's **max_fanout** capability. The default **fanout_load** value for a port is 0.0.

NOTE: Bidirectional ports are not included in maximum fanout calculations, so using the **set_fanout_load** command with a bidirectional port has no effect on optimization.

Use the **remove_attribute** or **reset_design** command to remove **fanout_load** values from ports. Use the **report_port** command to list fanout load values on ports.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following command sets a fanout load of 2 units on all output ports:

```
prompt> set_fanout_load 2 [all_outputs]
```

SEE ALSO

- all_outputs(2)
- compile(2)
- remove_attribute(2)
- report_port(2)
- reset_design(2)
- set_max_fanout(2)
- port_attributes(3)

set_fix_hold

Sets the **fix_hold** attribute on clocks in the current design.

SYNTAX

```
status set_fix_hold  
      clocks
```

ARGUMENTS

clocks

Specifies the clocks on which to set the **fix_hold** attribute.

DESCRIPTION

This command sets the **fix_hold** attribute on the specified clock objects in the current design. The **fix_hold** attribute informs optimization that hold time violations of the specified clocks should be fixed.

Fixing a hold violation requires slowing down data signals. Optimization fixes hold violations while fixing the design rules, but only if the maximum delay cost is not increased or if the **set_cost_priority** command is used to prioritize hold violations ahead of maximum delay cost. Optimization can violate setup and fix hold as long as the worst negative slack (WNS) is not made worse. Optimization might fix hold and violate setup on noncritical paths.

The **report_clock -attributes** command identifies the clocks that have the **fix_hold** attribute.

Use the **remove_attribute** command to remove the **fix_hold** attribute.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following command sets the **fix_hold** attribute on clock clk1.

```
prompt> set_fix_hold clk1
```

The following command removes the **fix_hold** attribute from clock clk1.

```
prompt> remove_attribute [get_clocks clk1] fix_hold
```

SEE ALSO

- compile(2)
- create_clock(2)
- remove_attribute(2)
- report_clock(2)
- reset_design(2)
- set_prefer(2)

set_fix_multiple_port_nets

Sets the **fix_multiple_port_nets** attribute to a specified value on the current design or a list of designs.

SYNTAX

```
int set_fix_multiple_port_nets  
  -default | -all  
  [-feedthroughs]  
  [-outputs]  
  [-constants]  
  [-buffer_constants]  
  [-no_rewire]  
  [-exclude_clock_network]  
  [design_list]
```

ARGUMENTS

-default

Removes the **fix_multiple_port_nets** attribute so that **compile** will use its default priority, that is no multiple port nets fixing.

-all

Inserts the same as **-feedthroughs -outputs -constants**. Note that logic constants are duplicated, not buffered. To fix constant port nets by buffering them, use the **-buffer_constants** option.

-feedthroughs

Inserts buffers to isolate input ports from output ports at all levels of the hierarchy.

-outputs

Inserts buffers so that no cell driver pin drives more than one output port at any level of the hierarchy.

-constants

Duplicates logic constants so that no constant drives more than one output port at any level of the hierarchy.

-buffer_constants

Buffers logic constants instead of duplicating them.

-no_rewire

Disables a rewiring attempt before buffering multiple port nets and constant port nets.

-exclude_clock_network

Excludes nets belonging to the clock network from multiple port net fixing.

design_list

Specifies a list of designs. The default is the current design.

DESCRIPTION

Sets the **fix_multiple_port_nets** attribute on the current design, unless a list of designs is specified.

When the command is used with a list of designs, the constraints of each design will be available when that design is made current by using the **current_design** command before running **compile**. The same way, **report_constraints** will only show the constraints of the current design.

This attribute controls whether **compile** inserts extra logic into the design to ensure that there are no feedthroughs, or that there are no two output ports connected to the same net at any level of hierarchy.

The default is not to add any extra logic into the design to fix such cases.

Certain three-state nets cannot be buffered, because this changes the logic functionality of the design.

If **set_fix_multiple_port_nets** is specified more than once on a design, the most recent setting is used and the earlier values are removed. To undo **set_fix_multiple_port_nets**, use the **-default** option, the **remove_attribute** command, or the **reset_design** command.

To view the current setting of the **fix_multiple_port_nets** attribute, use **report_compile_options**.

For some backward compatibility, if **compile** is run on a design with no **fix_multiple_port_nets** attribute and the obsolete variable **compile_fix_multiple_port_nets** is set to true, the attribute is set on the design to correspond with **set_fix_multiple_port_nets -all**. The attribute always has precedence over the variable setting. If **compile** is run when the attribute exists and has a different value than the variable, a warning is printed that the variable is being ignored. Consequently, if the choice of multiple port net fixing has to be changed between two **compile** commands on the same design, the **set_fix_multiple_port_nets** command must be used since changes to the **compile_fix_multiple_port_nets** variable after the first compile will be ignored.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the **fix_multiple_port_nets** attribute so that feedthroughs are buffered and constants are duplicated.

```
prompt> set_fix_multiple_port_nets -feedthroughs -constants
```

The following example sets the **fix_multiple_port_nets** attribute so that multiple outputs and constants are buffered.

```
prompt> set_fix_multiple_port_nets -outputs -buffer_constants
```

SEE ALSO

compile(2)
current_design(2)
remove_attribute(2)
reset_design(2)

set_flatten

Sets or removes the **flatten** attribute on specified designs or on the current design, to enable or disable the flattening optimization step during **compile**.

SYNTAX

```
int set_flatten  
    [true | false]  
    [-effort low | medium | high]  
    [-minimize single_output | multiple_output | none]  
    [-phase true | false] [-design design_list] [-quiet]
```

Data Types

design_list list

ARGUMENTS

true | false

When *true*, sets the **flatten** attribute on the designs in *design_list* and enables flattening for those designs. When *false* (the default value), the **flatten** attribute is removed and flattening is turned off for those designs.

-effort low | medium | high

Indicates the level of CPU effort that **compile** uses to flatten a design. The default is *low*.

-minimize single_output | multiple_output | none

Indicates the minimization strategy to be used after a design is flattened. If *single_output* (the default value), Design Compiler independently minimizes the equations for each output in a design. If *multiple_output*, Design Compiler minimizes all logic for a design. If *none*, Design Compiler does not perform minimization.

-phase true | false

When *true*, allows logic flattening to invert the phase of outputs during **compile**. When *false* (the default value), logic flattening does not invert the phase of outputs. This option is used only if the **flatten** attribute is set.

-design *design_list*

Specifies a list of designs to be flattened. The default is the current design.

-quiet

Indicates that warning messages are not to be displayed.

DESCRIPTION

Sets or removes the **flatten** attribute on specified designs or on the current design, enabling or disabling the flattening optimization step during **compile**. If no options are specified, the **flatten** attribute is set on the current design.

To enable flattening during optimization across the entire design hierarchy, issue the **set_flatten** command on the top design and each subdesign. To display the values of all **flatten** attributes, use **report_compile_options**.

This optimization step reduces a logic network to a two-level sum-of-products (AND/OR) representation by removing all intermediate variables. Full flattening eliminates all existing logic structure. By removing sub-optimal intermediate variables, **compile** is able to use other intermediate variables that otherwise might not have been found.

If phase assignment has been enabled by invoking the **set_flatten** command with **-phase true**, then the flattening optimization step in **compile** will compare the implementation of each logic equation in both its original and complemented form and choose the form with the smallest estimated area.

When the minimization strategy is set to *single_output*, Design Compiler minimizes the equations for each output individually. This results in the smallest implementation for each output, but the design as a whole might not be the most efficient because product (AND terms) are not well shared between outputs. When the strategy is *multiple_output*, Design Compiler minimizes all the logic for a design, and shares as many product terms between outputs as possible.

NOTE: Setting the **flatten** attribute can potentially cause **compile** to have long run times or to run out of memory, because logic flattening and equation minimization is a potentially exponential process. If you encounter an "out of memory" error or excessive run time, you can modify the options one at a time and re-run after each modification. Here is the recommended order for modifying the options:

1. Use *single_output* instead of *multiple_output* minimization strategy.
2. Use the **-phase false** option instead of the default **-phase true** option.
3. Use a lower effort setting.

To remove the **flatten** attribute, use **remove_attribute** or execute **set_flatten false**. **reset_design** removes **all** attributes, including **flatten**.

For more details on flattening during optimization, refer to the *Design Compiler Reference Manual: Optimization and Timing Analysis*.

EXAMPLES

In this example, **flatten** is enabled on the design "TEST".

```
prompt> set_flatten -design TEST
```

In this example, **flatten** is enabled on the current design with the **effort** and **minimize** options specified, and **flatten** is disabled on the design "OLD."

```
prompt> read TEST
prompt> set_flatten -effort high -minimize multiple_output
prompt> set_flatten -design OLD false
```

SEE ALSO

compile(2)
current_design(2)
remove_attribute(2)
reset_design(2)
attributes(3)

set_fp_base_gate

Sets either a library leaf cell area or a user-specified cell area as the base unit area to use for gate equivalence calculations related to estimating the size of black boxes.

SYNTAX

```
status set_fp_base_gate
{-cell master_name | -area cell_area}
```

ARGUMENTS

-cell *master_name*

Specifies the master name of the library leaf cell to use as the base unit area for gate equivalence calculations.

-area *cell_area*

Specifies the cell area in square microns to use as the base unit area for gate equivalence calculations.

DESCRIPTION

Sets either a library leaf cell area or a user designated cell area as the base unit area to use for gate equivalence calculations related to estimating the size of black boxes.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

Set base gate area to cell master UNIT

```
prompt> set_fp_base_gate -cell UNIT
```

Set base gate area to 10 square microns

```
prompt> set_fp_base_gate -area 10
```

SEE ALSO

`estimate_fp_black_boxes(2)`

set_fsm_encoding

Specifies the bit encodings for states in the current design.

SYNTAX

```
int set_fsm_encoding
    encoding_list
```

Data Types

encoding_list list

ARGUMENTS

encoding_list

Specifies a list of all the states in the current design with the assigned bit encodings. Each state name and encoding must be enclosed in double quotation marks. (" ").

Encodings are specified using one of two formats. The first format consists of a base specifier followed by the pound (#) character, followed by a string of digits in the given base numbering system. The string of digits can be separated by an optional underscore character to make the encodings easier to read. If the base specifier and pound (#) character are omitted, the string of digits are interpreted as a decimal (base 10) value. For example, 15 can be entered in this format as:

```
2#1111
8#17
10#15
15
16#f
```

The second format consists of the caret (^) character, followed by a character base specification, followed by a string of digits in the given base. For example, 15 can be entered in this format as:

```
^B1111 (Binary)
^O17 (Octal)
^D15 (Decimal)
^Hf (Hexadecimal)
```

If the list contains more than one state, the list of states must be enclosed in braces ({}).

DESCRIPTION

The **set_fsm_encoding** command specifies the set of all legal states and state encodings in the current design.

Only one *encoding_list* can be active at a time on the current design.

EXAMPLE

The following example assigns codes to two states in the machine. The encodings are removed for states not in the list.

```
prompt> set_fsm_encoding { "IDLE=2#000" "FIVE=2#001" }
```

SEE ALSO

`compile(2)`

`set_fsm_encoding_style(2)`

set_fsm_encoding_style

Defines the encoding style for assigning unencoded states.

SYNTAX

```
int set_fsm_encoding_style  
  {one_hot | zero_one_hot | binary | gray | auto | neutral}
```

ARGUMENTS

{one_hot | zero_one_hot | binary | gray | auto | neutral}

Specifies the encoding style that is to be used during state assignment. The encoding styles are described in the DESCRIPTION section. The default is **auto**.

DESCRIPTION

Defines the encoding style to be used for determining unencoded states during the generation of a state machine design. (The process of determining encodings for states is referred to as state assignment.) If you have manually specified encodings for any of the states in the design, state assignment does not reassign any of these encodings. The code length is based on 1) the length of any manually-assigned encodings, or 2) the number of instance names specified in the **set_fsm_state_vector** command. If the code length cannot be determined by any of these criteria, it is determined by the encoding style selected. **one_hot** encoding style can be re-implemented as **zero_one_hot** because of register constant removal or register merging.

The encoding styles are described below.

The **one_hot** encoding style generates codes with a bit length equal to the number of states in the state machine, each state being represented by one bit position. The code is a string of zeros, interrupted by a one in the state's particular bit position. A warning is generated if the number of elements in the state vector is not equal to the number of states, in which case the length of the code is reset to the number of states in the machine. If the **one_hot** encoding style is specified, all manually-assigned codes must also be **one_hot** codes.

The **binary** and **gray** encoding styles sequentially assign codes to unassigned states using a binary numbering or gray numbering sequence, respectively. The binary numbering sequence consists of using binary values to encode the sequence of counting integers. For example, four states encoded using two bits would be numbered 0, 1, 2, and 3, represented by the binary encodings of 00, 01, 10, and 11. The **gray** numbering sequence assigns codes to states such that successive codes never differ by more than one bit. In this case, the same four states encoded using the gray encoding style would be numbered 0, 1, 3, and 2, represented by the encodings of 00, 01, 11, and 10. Note that in this encoding sequence only one bit position changes value in going from one to the next (for example, 01 to 11.) The sequence in which the codes are assigned to the states is based on the ordering of the states. States are ordered via their entry in the state table, or may be manually assigned using the **set_fsm_order** command. If any states have manually assigned encodings, then these states and their respective codes are omitted from the ordering of binary or gray

codes. Additionally, if any state is unordered, an arbitrary ordering is used that does not affect any existing order. The default code length is base log2 of the number of states in the machine.

The **auto** encoding style generates codes chosen in a manner to best minimize the logic of the state machine. If the length of the bit encodings cannot be determined using the criteria specified below, the Design Compiler's **auto** encoding style selects the code length that best minimizes the state machine.

The **neutral** encoding style will keep state assign as user defined in the HDL code, i.e., keep the encoding style unchanged.

EXAMPLES

The following example shows four states, "IDLE", "FIVE", "TEN", and "OWE_DIME", with a **one_hot** encoding style:

```
prompt> set_fsm_encoding_style one_hot
```

During **compile**, the state assignment encodes the states as 1000, 0100, 0010, and 0001, respectively.

The following example shows the same machine with the states ordered as "IDLE", "FIVE", "TEN", and "OWE_DIME", with a **binary** encoding style:

```
prompt> set_fsm_encoding_style binary
```

During **compile**, the state assignment encodes the states as 00, 01, 10, and 11, respectively.

SEE ALSO

- `compile(2)`
- `current_design(2)`
- `set_fsm_order(2)`
- `set_fsm_state_vector(2)`

set_fsm_minimize

Determines whether or not state minimization is to be performed on the state machine design during **compile**.

SYNTAX

```
int set_fsm_minimize  
    true | false
```

ARGUMENTS

true | false

Indicates whether or not state minimization is to be performed. The default is *false*.

DESCRIPTION

Determines whether or not state minimization is to be performed on the state machine design during **compile**. State minimization is not performed again on a design that has already had state minimization performed (for example, with the **fsm_minimize** command). You can additionally specify states that are to be preserved by using the **set_fsm_preserve_state** command.

EXAMPLES

The following examples show a state table design being read in, set up for state minimization, and **compiled**.

```
prompt> read -f st example.st  
prompt> set_fsm_minimize true  
prompt> compile
```

SEE ALSO

`compile(2)`
`set_fsm_preserve_state(2)`

set_fsm_order

Sets the ordering of states in a state machine design.

SYNTAX

```
int set_fsm_order
    state_list
```

Data Types

state_list list

ARGUMENTS

state_list

List of states in the state machine to which an order is assigned. All states in the *state_list* must be states in the state machine. If more than one state is specified, they must be enclosed in braces ({}).

DESCRIPTION

Defines an ascending ordering of states in the current design. Any previously-assigned order is removed and replaced by the new order specified in *state_list*. Any states that are not included in *state_list* are given an arbitrary order that does not violate the specified ordering.

compile uses state ordering when determining encodings for unencoded states using the **binary** or **gray** encoding style. (See **set_fsm_encoding_style** for a description of those encoding styles.) The first state is assigned the lowest unused state code, and each successive state is assigned the next unused code in the **binary** or **gray** numbering sequence.

An empty *state_list* removes the ordering of all states.

EXAMPLES

The following example defines the ordering of states such that "IDLE" is the first state, "FIVE" is the second state, and so on:

```
prompt> set_fsm_order { IDLE FIVE TEN OWE_DIME }
```

To remove the ordering from all states, enter an empty *state_list*:

```
prompt> set_fsm_order { }
```

SEE ALSO

`compile(2)`

`set_fsm_encoding_style(2)`

set_fsm_preserve_state

Specifies states to be preserved during state minimization.

SYNTAX

```
int set_fsm_preserve_state  
    state_list
```

Data Types

state_list list

ARGUMENTS

state_list

List of state names to be preserved. If more than one state is specified, they must be enclosed in braces ({}).

DESCRIPTION

Specifies those states which are to be preserved during state minimization.

This command overrides any previously specified state preservation. To undo this command, enter **set_fsm_preserve_state** with an empty *state_list*.

EXAMPLES

The following command specifies that state "IDLE" is to be preserved during state minimization:

```
prompt> set_fsm_preserve_state { IDLE }
```

SEE ALSO

set_fsm_state_vector

Specifies the instance names for flip-flops used to implement the state vector.

SYNTAX

```
int set_fsm_state_vector  
    vector_list
```

Data Types

vector_list list

ARGUMENTS

vector_list

Specifies a list of instance names for the state vector in the state machine. If more than one name is specified, they must be enclosed in braces ({}). The first instance name in the list is considered to be the most significant bit of the vector, while the last instance name is considered the least significant bit of the vector.

DESCRIPTION

Explicitly names the instances of the flip-flops used to store the current state of the state machine. Additionally, the specification of a state vector defines the length of the codes used for the encoded states in the machine.

EXAMPLES

The following example specifies the state vector's instance names:

```
prompt> set_fsm_state_vector { ff0 ff1 ff2 ff3 }
```

SEE ALSO

compile(2)

```
set_fsm_encoding(2)
```

set_gui_stroke_binding

Set the command binding for a stroke.

SYNTAX

```
set_gui_stroke_binding  
dictionary_name  
stroke_sequence  
[-builtin builtin_cmd_name]  
[-clear]  
[-tcl_cmd tcl_command]  
[-label tcl_command_label]
```

ARGUMENTS

dictionary_name

Specify the dictionary to set the binding into. Windows that support stroke commands have one or more dictionaries to use when evaluating stroke commands.

stroke_sequence

Specifies the sequence of grids that defines the path of the stroke. See the *Stroke Sequences* section, below, for more information.

-builtin *builtin_cmd_name*

Specifies the name of a built-in command option to be executed.

-clear

Clears the existing command for the specified stroke sequence.

-tcl_cmd *tcl_command*

Specifies the Tcl command to be executed when the stroke is entered.

-label *tcl_command_label*

Specifies the menu label for the tcl command. This menu is displayed for line and snap_line strokes after a delay to provide information on the current bindings.

DESCRIPTION

Select a stroke by specifying a symbol with the mouse. The location of the down-click starts the stroke, and the path that the mouse follows while the mouse button is pressed determines the stroke that is entered. This stroke is mapped onto a numbered 3x3 grid and is transformed into a series of these grid numbers. A stroke dictionary is used to map this sequence of numbers to the command to be executed. When you release the mouse button, this command is executed.

You can use the **report_gui_stroke_builtins** command to determine the available built-in (non-Tcl) commands. You can use the **report_gui_stroke_bindings** command to determine the existing bindings.

Stroke Types

The stroke command facility supports the following modes for entering strokes: line-based, rectangle-based, and path-based. All stroke entry types generate stroke bindings that are the same.

If you are an occasional user, the line-based and rectangle-based entry modes are easier to specify, but they limit the number of strokes that can be generated. If you are an advanced user, the path-based entry mode, which is more difficult to specify, allows many more strokes to be generated.

See the man page for the **set_gui_stroke_preferences** command for complete information on stroke entry types,

Stroke Sequences

The path of a stroke is mapped onto a 3x3 grid. The size of the grid squares are determined to ensure that the stroke covers the width or height of the grid. The stroke is centered in the grid of horizontal or vertical strokes.

The grid is numbered as shown below:

```
1 2 3
4 5 6
7 8 9
```

The stroke is mapped onto the grid and converted to a sequence of these grid numbers, which is called a *stroke sequence*. For example, a diagonal stroke from the top-left corner of the grid to the bottom-right corner has a stroke sequence of **159**; a left-to-right horizontal stroke has a stroke sequence of **456**.

Since the strokes are path-based, a single grid might occur more than once in a sequence. For example, a stroke that moves horizontally left-to-right and right-to-left has a stroke sequence of **12321**.

Strokes also support the Shift, Control, and Alt modifier keys. If one or more of these modifiers are depressed when the stroke is started, they are applied to the stroke. The modifiers are prepended to the stroke sequence, with **s** delimiting Shift, **c** delimiting control, and **a** specifying Alt. The modifiers are separated from the sequence by using a colon (:). For example, a left-to-right horizontal stroke sequence is **123**, if unmodified; **s:123** with Shift; **c:123** with Control; and **sc:123** with Shift-Control modifiers.

A click is supported as a special (degenerate) case of a stroke. If the mouse down-click and mouse up-click event happen close enough together (in both time and location), the stroke is recognized as a click and a stroke sequence of **5** is generated.

The stroke facility also supports the definition of a default command binding that is to be used for any bindings not explicitly set. This default binding is specified by using a simple stroke sequence of **0**. If no binding is registered for a given set of modifiers and sequence, the tool searches for a stroke binding with the same modifiers and a sequence of **0** and uses that if it is found.

Built-In Commands

The application supporting stroke commands might support operations that are not available from the Tcl command language to be invoked via the stroke command. These commands are given a name, and you can use the **-builtin** option to specify a stroke binding to invoke these commands

The stroke facility always supports the built-in commands for zooming and panning listed below.

Zoom_In

Zoom in by a fixed percentage.

Zoom_Out

Zoom out by a fixed percentage

Zoom_In_Rect

Zoom in to fix the rectangle specified by the bounding box for the stroke.

Zoom_Out_Rect

Zoom out centering on the rectangle specified by the bounding box for the stroke.

Zoom_Full

Zoom and pan to fit the entire drawing centered in the window.

Pan_Center_Rect

Pan to center the rectangle specified by the bounding box for the stroke.

Tcl Command Bindings

The bindings support invoking Tcl-based commands to allow the functionality of the mechanism to be expanded. Tcl-based commands are allowed to contain key values that are substituted with data from the command before they are executed by the interpreter, thus allowing these commands to have access to the context information for the stroke. A leading percent (%) character specifies a key value..

The keys listed below are supported by tcl commands.

%rect

Substitutes the coordinates for the bounding box of the stroke. The value is a list of points $\{\{x1\ y1\} \{x1\ y2\}\}$, where you substitute bounding box coordinates for $x1$, $y1$, $x2$, and $y2$.

%startPoint

Specifies the starting point of the stroke. The value is in the form $\{x\ y\}$, where you substitute starting point coordinates for x and y .

%endPoint

Specifies the ending point of the stroke. The value is in the form $\{x\ y\}$, where you substitute starting point coordinates for x and y .

%sequence

Specifies the stroke sequence that invoked this command.

%modifiers

Specifies the modifiers of the stroke sequence that invoked this command.

EXAMPLES

The following example defines a left-to-right horizontal binding to execute the built-in operation that centers the location of the stroke in the viewport.

```
psyn_gui-> set_gui_stroke_binding Graphics 123 \
```

-builtin Pan_Center_Rect

The following example calls a tcl command for the Shift-modified, lower-left to upper-right stroke sequence.

```
psyn_gui-t> set_gui_stroke_binding Graphics \  
-tcl s:753 my_tcl_command
```

The following example calls a tcl command taking a rectangle as an argument for the Shift-control, lower-left to upper-right stroke sequence.

```
psyn_gui-t> set_gui_stroke_binding Graphics \  
ss:753 -tcl {my_tcl_command %rect}
```

SEE ALSO

report_gui_stroke_bindings(2)
report_gui_stroke_builtins(2)
set_gui_stroke_preferences(2)

set_gui_stroke_preferences

Set preferences controlling stroke command entry.

SYNTAX

```
set_gui_stroke_preferences  
[-type stroke_entry_type]  
[-shift]  
[-ctrl]  
[-alt]  
[-extended_help_delay ms_delay]
```

ARGUMENTS

-type *stroke_entry_type*

Specifies the are **snap_line**, **line**, **rect**, **snap_path**, and **path**.

-shift

Sets the type for the shift modifier. This option is valid only when using the **-type** option.

-alt

Sets the type for the alt modifier. This option is valid only when using the **-type** option.

-ctrl

Sets the type for the ctrl modifier. This option is valid only when using the **-type** option.

-extended_help_delay *ms_delay*

Specifies the number of milliseconds to wait before showing the extended help (menu) for a stroke. A value of less than 500 makes the extended feedback immediate. Extremely large values suppress the extended feedback almost completely.

DESCRIPTION

Select a stroke command by specifying a symbol with the mouse. The location of the down-click starts the stroke, and the path that the mouse follows while the mouse button is pressed determines the stroke that is entered. This stroke is mapped onto a numbered 3x3 grid and is transformed into a series of these grid numbers. A stroke dictionary is used to map this sequence of numbers to the command to be executed. When you release the mouse button, this command is executed.

Use this command to control the user interface behavior when you are specifying a stroke. The preferences allow the strokes to be

used by both application experts and occasional users of the application.

The two primary types of stroke entry mechanisms are the line-based type and the stroke type. The primary preference supported is the stroke type. A line-based type determines the stroke based on the values of the stroke's start and end points. The path between the points is ignored. A line-based stroke turns the stroke facility into a form of "pie-menu" that supports nine different menu items. A path-based type determines the stroke based on the path between the stroke's start and end points. This allows an extremely large number of unique strokes to be defined, and it also requires a significantly-higher amount of precision when specifying the stroke than when using the line-based type.

The snap_line, line, and rect stroke types are all line-based strokes: the snap_path and path are path-based strokes.

Stroke Types

snap_line

The line between the start and end points is snapped onto the nearest 45 degree angle. It allows the generation of eight different stroke sequences and the click stroke sequence.

line

The line between the start and end points specifies the stroke. It allows the generation of eight different stroke sequences and the click stroke sequence.

rect

The rectangle between the start and end points specifies the stroke. It allows the generation of four different stroke sequences and the click stroke sequence.

snap_path

This path-based stroke snaps the segments of the path onto the nearest 45 degree angle to make it easier to generate stroke sequences with diagonals and straight lines. Since it is a path-based stroke, the number of unique sequences supported for this entry type is extremely large.

path

This path-based stroke is a free-formed path from the start point to the end point. Since it is a path-based stroke, the number of unique sequences supported for this entry type is extremely large.

The unmodified stroke type is the default type for all modifiers when a type has not been specified for the modifier. Different stroke types can be used for each combination of modifiers by setting the stroke type for the modifier with this command.

EXAMPLES

The following example sets the stroke type to snap_line.

```
psyn_gui-> set_gui_stroke_preferences -type \  
snap_line
```

The following example sets the stroke type for Shift-Control modified strokes to snap_line.

```
psyn_gui-> set_gui_stroke_preferences -shift \  
-ctrl -type snap_line
```


SEE ALSO

report_gui_stroke_bindings(2)
report_gui_stroke_builtins(2)
set_gui_stroke_binding(2)

set_host_options

Controls the maximum number of CPU cores that can be used for parallel execution.

SYNTAX

```
status set_host_options  
-max_cores number_of_cores
```

Data Types

number_of_cores integer

ARGUMENTS

-max_cores *number_of_cores*

Specifies the maximum number of CPU cores that are allowed for parallel execution.

The **-max_cores** option checks the number of CPU cores available on the execution host. If you specify a higher number of CPU cores than are available, the tool limits the maximum CPU cores to the number of available cores and issues a UIO-230 warning message.

The **-max_cores** option also checks the load on the execution host. If the host is heavily loaded, the tool uses fewer cores than requested to avoid an overall system slow down. In this event, the tool issues a UIO-231 warning message.

To disable these checks and use the number of CPU cores you requested, set the **disable_multicore_resource_checks** variable to **true**. However, this setting can slow down the tool if the host does not have the necessary cores or is heavily loaded.

DESCRIPTION

The **set_host_options** command is used to specify the maximum number of CPU cores that the tool can use during parallel execution. The default is 1, which indicates that parallel execution is not enabled. The maximum number of cores that can be specified is 16.

To disable parallel execution, set the **-max_cores** option to 1 or run the **remove_host_options** command.

EXAMPLES

The following example tells the tool that it can use up to four cores for parallel execution:

```
prompt> set_host_options -max_cores 4
```

SEE ALSO

- compile_ultra(2)
- remove_host_options(2)
- report_host_options(2)
- disable_multicore_resource_checks(3)
- UIO-230(n)
- UIO-231(n)

set_hpc_options

Applies high performance core settings to the current design

SYNTAX

```
status set_hpc_options  
  [-list]  
  [-core core_name]  
  [-stage stage_name]  
  [-report_only]
```

ARGUMENTS

-list

Lists the supported core names and stage names

-core *core_name*

Specifies the high performance core. Use the **-list** option to view the supported core names

-stage *stage_name*

Specifies the stage name. Use the **-list** option to view the supported stage names

-report_only

Reports only the core and technology-node settings and does not any design constraints

DESCRIPTION

This command applies high performance core settings to the current design. It sets the parameters and application options for synthesis, placement, and route engines for better performance and power. When you use the **set_hpc_options** command, all the controls are applied to the design and the "hpc_core" attribute is also set on the design.

You must use the **set_technology** command before using the **set_hpc_options** command.

You can use this feature in Design Compiler NXT topographical mode.

EXAMPLES

In following example, the **set_hpc_options** command lists the cores and stages that are supported in the tool:

```
prompt> set_hpc_options -list  
cores: A55 A76  
stages: compile compile_inc
```

SEE ALSO

set_technology(2)

set_icc2_options

Specifies the options used to launch IC Compiler II from within Design Compiler Graphical or DC Explorer physical mode.

SYNTAX

```
int set_icc2_options
  -icc2_executable executable
  -ref_libs       library
  [-check]
  [-work_dir      directory]
  [-golden_upf     upf_file]
  [-technology     technology_file
  | -use_technology_lib tech_lib_name]
  [-keep_files]
  [-reset]
  [-convert_sites convert_sites]
  [-scale_factor  scale_factor]
  [-congestion_use_global_route {true | false}]
```

Data Types

```
executable    string
library       string
directory     string
upf_file      string
technology_file string
scale_factor  int
```

ARGUMENTS

-icc2_executable *executable*

Specifies the path to the IC Compiler II executable that is used to invoke IC Compiler II from within Design Compiler Graphical or DC Explorer physical mode. By default, Design Compiler uses the `icc2_shell` executable specified by the `$PATH` variable.

-ref_libs *library*

Specifies the IC Compiler II reference libraries that are used when invoking IC Compiler II. This is a required option.

-check

Performs the startup checks and reports any errors that would prevent the **start_icc2** command from successfully launching IC Compiler II.

-work_dir *directory*

Specifies the directory where the temporary files are stored when IC Compiler II is launched. Enter the name of your working directory with the relative path from the current directory. If you do not provide the path to a specific directory, the files are stored in the directory created at the `$TMPDIR/dcg_unique_string` location if you specified a location with the **TMPDIR** UNIX environment variable. If you did not specify a location with the **TMPDIR** variable, the files are stored in the `/tmp` directory location.

-golden_upf upf_file

Specifies the golden UPF file that is used in IC Compiler II. This is a required option in the golden UPF flow.

-technology technology_file

Specifies the technology file used by the **create_lib** command in IC Compiler II. This option is required with the exception of **start_icc2** command in Milkyway mode. If you do not provide the technology file, the **start_icc2** command writes out the technology file that will be used by IC Compiler II. In NDM mode, this option is not required and will be ignored.

-use_technology_lib tech_lib_name

Specifies the reference library to use as a dedicated technology library. The library must contain a technology section and must be one of the libraries specified with the **-ref_libs** option. If the library doesn't contain a technology section, **set_icc2_options** command will not fail but **create_lib** command in IC Compiler II will fail.

This option is mutually exclusive with the **-technology** option, and it must be used with the **-ref_libs** option.

-keep_files

Prevents deletion of the temporary files that are created during and after the launch of the IC Compiler II session.

-reset

Resets the options specified to invoke IC Compiler II.

-convert_sites convert_sites

Specifies the mapping of DEF site names using the following syntax: `{ {from_site to_site} ...}` This option converts all DEF rows of site from_site to rows of site to_site in IC Compiler II.

-scale_factor scale_factor

You can use this option only in the Milkyway mode to create IC Compiler II design library using IC Compiler II Link. It specifies the length precision for this library. The length precision for the library and all its reference libraries must be identical. The value is specified in terms of units per micron. By default, a length precision of 10000 is used, which implies one internal unit is equal to one Angstrom or 0.1 nm.

Make sure to set a scale factor value that results in a whole number of database units (dbu) per minimum grid spacing. For example, if your minimum grid spacing is 1 nm, you can use the default scale factor value of 10000 dbu per micron.

-congestion_use_global_route {true | false}

If this option is set to **true**, **report_congestion** command computes and displays congestion statistics for the current design using IC Compiler II. If you also set the **placer_enable_enhanced_router** variable to **true** along with this option, the tool enables Z-route-based congestion driven placement using IC Compiler II. This option is only available in Design Compiler NXT.

DESCRIPTION

The **set_icc2_options** command specifies the options used to launch IC Compiler II from within Design Compiler Graphical or DC Explorer physical mode. After specifying the **set_icc2_options** command settings, run the **start_icc2** command to begin floorplanning using IC Compiler II.

You can also use the **report_icc2_options** command to report the **set_icc2_options** settings.

EXAMPLES

The following example sets the IC Compiler II libraries and executable to invoke IC Compiler II from within Design Compiler Graphical:

```
prompt> set_icc2_options -ref_libs "lib1.ndm lib2.ndm" \
-icc2_executable /my_executable_location/icc2_shell
1
```

The following example sets the technology library

```
prompt> set_icc2_options -use_technology_lib technology_lib.nlib
Error: Cannot specify option '-use_technology_lib' of command 'set_icc2_options' without '-ref_libs'. (DCT-310)
0
```

```
prompt> set_icc2_options -use_technology_lib technology_lib.nlib \
-ref_libs { lib1.nlib }
Error: Technology library '%s' does not match any library on the reference library list. (DCT-311)
0
```

```
prompt> set_icc2_options -use_technology_lib technology_lib.nlib \
-ref_libs { technology_lib.nlib }
1
```

```
prompt> set_icc2_options -use_technology_lib ../nlibs/technology_lib.nlib \
-ref_libs { ../nlibs/technology_lib.nlib }
1
```

```
prompt> set_icc2_options -use_technology_lib technology_lib.nlib \
-ref_libs { ../nlibs/technology_lib.nlib }
1
```

The following example performs startup checks and issues an error due to the nonexistent IC Compiler II executable:

```
prompt> set_icc2_options -ref_libs "lib1.ndm" \
-icc2_executable my_path/icc2_shell -check
Error: Could not find the IC Compiler II executable my_path/icc2_shell. (DCT-224)
0
```

The following example stores the temporary files generated during the launch of the IC Compiler II session in the my_work_dir location and does not delete them once you exit the IC Compiler II session:

```
prompt> set_icc2_options -ref_libs "lib1.ndm" \
-work_dir my_work_dir -keep_files
1
```

SEE ALSO

compile_ultra(2)


```
start_icc2(2)  
report_icc2_options(2)
```

set_icc_dp_options

Specifies the options used to invoke floorplan exploration from Design Compiler Graphical.

SYNTAX

```
int set_icc_dp_options  
  -work_dir directory  
  -icc_executable executable  
  -check flag  
  -file_name_prefix prefix  
  -keep_files flag
```

Data Types

```
directory  string  
executable string  
flag       flag  
prefix     string
```

ARGUMENTS

-work_dir *directory*

Specifies the directory where the temporary files are stored during floorplan exploration. Enter the name of your working directory with the relative path from the current directory. If you do not specify the **-work_dir** option, the tool stores the temporary files in the directory created at the \$TMPDIR/dcg_*unique_string* location if you specified a location by using the **TMPDIR** UNIX environment variable. If you did not specify a location with the **TMPDIR** variable, the tool stores the files in the /tmp directory.

-icc_executable *executable*

Specifies the path to the IC Compiler executable that is used for floorplan exploration. By default, Design Compiler uses the `icc_shell` executable specified by the **\$PATH** variable.

-check *flag*

Performs the startup checks and issues errors, if any.

-file_name_prefix *prefix*

Specifies the prefix name for any generated files, such as the floorplanning scripts and floorplan files. The default file prefix naming style is %s_%d where %s is the design name and %d is the process ID. You can provide any format using these two directives. See the Examples section.

-keep_files *flag*

Specifies not to delete the temporary files that are created during and after the launch of the floorplan exploration session.

DESCRIPTION

This command specifies the options used to launch Design Compiler Graphical floorplan exploration. After setting **set_icc_dp_options**, use the **start_icc_dp** command to begin floorplanning. You can also use the **report_icc_dp_options** command to report the **set_icc_dp_options** settings.

EXAMPLES

```
prompt> set_icc_dp_options \  
-work_dir /tmp/work_dir \  
-icc_executable /your/location/icc_shell
```

```
prompt> set_icc_dp_options \  
-check
```

```
prompt> set_icc_dp_options \  
-file_name_prefix "temp_cel"
```

```
prompt> set_icc_dp_options \  
-file_name_prefix "Tryrun1"
```

```
prompt> set_icc_dp_options \  
-file_name_prefix "Tryrun%s"
```

SEE ALSO

compile_ultra(2)
report_icc_dp_options(2)
start_icc_dp(2)

set_ideal_latency

Specifies ideal network latency.

SYNTAX

```
status set_ideal_latency
  [-rise | -fall]
  [-min | -max]
  delay
  object_list
```

Data Types

```
delay      float
object_list list
```

ARGUMENTS

-rise | -fall

Specifies whether the latency is for data rise or data fall transition. If you do not specify **-rise** or **-fall**, both values are set. The **-rise** and **-fall** options are mutually exclusive.

-min | -max

Specifies whether the latency is to be used for minimum delay analysis or maximum delay analysis. By default, the delay is used for both maximum and minimum delay analysis. The **-min** and **-max** options are mutually exclusive.

delay

Specifies the ideal latency value on pins in the ideal network. The delay must be expressed in units consistent with the technology library used during optimization. For example, if the technology library specifies delay values in nanoseconds, *delay* must be expressed in nanoseconds.

object_list

Specifies a list of leaf cell pins or top-level ports that are the startpoints of the timing arcs for which ideal latency is to be set.

DESCRIPTION

Sets the ideal latency on top-level ports and leaf cell pins of the ideal network.

Design Compiler assumes ideal timing for ideal networks and ideal nets, which means pins have a specified ideal latency (from the **set_ideal_latency** command) or zero ideal latency by default. The ideal network is normally used for pre-layout to reduce runtime by avoiding unnecessary DRC optimization and retiming. Ideal latency provides an estimate of the ideal network or ideal nets for pre-layout.

The specified latency value overrides the internally-estimated cell delay value and net delay value. If the specified pins do not belong to the ideal network, ideal nets, or auto disable drc nets, the **report_ideal_network** command generates an error message and the *delay* value is not used for those pins.

The **set_ideal_latency** command affects all pins in the transitive fanout of the pins or ports. The total ideal latency at an ideal boundary pin is the sum of latency on the ideal network source and all ideal latencies on the path.

You can use **set_ideal_latency** for pins at lower levels of the design hierarchy. Pins are specified in the form of "INSTANCE1/INSTANCE2/PIN_NAME."

To list ideal latency values, use the **report_ideal_network** command.

To remove ideal latency values from a design, use the **remove_ideal_latency** or **reset_design** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example specifies a rise latency of 1.2 and a fall latency of 0.9 for the ports named A, B, and C.

```
prompt> set_ideal_latency 1.2 -rise {A B C}
prompt> set_ideal_latency 0.9 -fall {A B C}
```

SEE ALSO

- `remove_ideal_latency(2)`
- `remove_ideal_network(2)`
- `remove_ideal_transition(2)`
- `report_ideal_network(2)`
- `report_timing(2)`
- `set_ideal_net(2)`
- `set_ideal_network(2)`
- `set_ideal_transition(2)`

set_ideal_net

Sets the specified nets as ideal. Note that this command has been deprecated; you should use the **set_ideal_network -no_propagate** command instead.

SYNTAX

```
status set_ideal_net
      nets
```

Data Types

nets collection

ARGUMENTS

nets

Specifies the nets on which to set the **ideal_net** attribute. These nets must be visible from the current design.

DESCRIPTION

This command sets the specified nets as ideal nets. Note that this command has been deprecated; you should use the **set_ideal_network -no_propagate** command instead.

The global driver pins of the specified nets are marked as sources of an ideal network. These nets must be visible in the current design. The tool treats a net as an ideal net if all global driver pins of the net are ideal. By default, the tool treats all clock nets as ideal nets. However, this does not include the logic gates in the networks, and the nets driven by those logic gates. The ideal properties of the net's global driver pins do not propagate through logic gates. However, they propagate through hierarchies. Setting the global driver pins as ideal enables the ideal properties on all nets that are electrically connected to those pins.

To prevent clock nets from being treated as ideal nets, use the **set_auto_disable_drc_nets -clock false** or **set_propagated_clock [all_clocks]** command. After you run the **set_propagated_clock [all_clocks]** command, the **ideal_net** attribute is not reported by the **report_net** command.

Ideal nets are networks of nets that are free from maximum capacitance, maximum fanout, and maximum transition design rule constraints. Ideal nets are useful for reducing DRC violations caused by clock trees, because these networks usually have large maximum capacitance and maximum fanout violations. Ideal nets use the ideal latency specified by the **set_ideal_latency** command and ideal timing specified by the **set_ideal_transition** command for timing calculation. By default, zero ideal latency and zero transition are used. To automatically spread ideal attributes through a network, use the **set_ideal_network** command without the **-no_propagate** option.

The tool automatically sets the **size_only** attribute on the cells of the ideal network sources. In addition, an ideal net implies that this net is **dont_touch**. Note that the tool might optimize away combinational cells in the fanout of the ideal net. However, it guarantees that the ideal network sources are never lost. To automatically set clock, constant, or scan nets in the design as ideal, use the **set_auto_disable_drc_nets** command.

Use the **remove_ideal_net** command to remove the attributes set by the **set_ideal_net** command in the current design. Use the **set_auto_disable_drc_nets -none** command to remove the **auto_disable_drc_net** attribute. The **reset_design** command removes all attributes from the design, including both the ideal attributes and the **auto_disable_drc_net** attributes; however, by default, the tool still treats clock networks as ideal nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the **ideal_net** attribute on all nets in the current design:

```
prompt> set_ideal_net [get_nets]
```

SEE ALSO

- `remove_ideal_net(2)`
- `reset_design(2)`
- `set_auto_disable_drc_nets(2)`
- `set_ideal_latency(2)`
- `set_ideal_network(2)`
- `set_ideal_transition(2)`
- `set_dont_touch(2)`
- `set_dont_touch_network(2)`

set_ideal_network

Marks a set of ports or pins in the current design as sources of an ideal network. No delay computation is performed. However, the arrival time and required information is propagated in the transitive fanout of the ideal source.

SYNTAX

```
integer set_ideal_network  
  object_list  
  [-dont_care_placement]  
  [-no_propagate]
```

Data Types

object_list list

ARGUMENTS

object_list

Specifies a list of objects (ports, pins, or nets) to mark as the sources of an ideal network. If more than one object name is specified, each must be enclosed in quotation marks or braces {}. The source objects are input ports on the design or any internal pin, except for pins at hierarchical boundaries. If nets are provided, all of the nets' global driver pins are marked as ideal network sources, and the corresponding ideal network attributes are actually set on all of the global driver pins of the specified nets. These objects must be visible from the current design.

The **set_ideal_network** command accepts nets only when you specify the **-no_propagate** option.

-dont_care_placement

Indicates that the ideal network is not considered in placement. The nets in the ideal network are treated as disconnected. Random locations are assigned to ideal network cells. By default, the ideal network is placed at the lowest priority.

-no_propagate

Indicates that the ideal network is not propagated through logic gates, but it still propagates through hierarchies. Ideal properties are enabled on all nets that are electrically connected to ideal network sources. By default, this option is off.

DESCRIPTION

This command marks a set of ports or pins in the current design as sources of an ideal network. Ideal networks are an extension of ideal nets that incorporate automatic propagation of the **ideal** attribute. You specify only the source of the network; the **compile** command treats all nets, cells, and pins on the transitive fanout of these objects as ideal. The ideal property is automatically spread

by the tool and respread as necessary during **compile** optimizations. The criteria for propagating the ideal property, starting at the source pins and ports, are as follows:

- A pin is marked as ideal if you specify it by using the **set_ideal_network** command if it is either a driver pin and its cell is ideal or it is a load pin attached to a net that is ideal.
- A net is marked as ideal if all of its driving pins are ideal.
- A combinational cell is marked as ideal if all of its input pins are either ideal or attached to a constant net (and other input pins are ideal). Objects with the **case analysis** attribute set are not treated as constant.

Propagation traverses through combinational cells but stops at sequential cells. If an ideal network overlaps a clock network, the clock timing overrides the ideal timing for the clock part of the network.

In addition to disabling timing updates and timing optimizations, all cells and nets in the ideal network have the **dont_touch** attribute set.

The **size_only** attribute is set on all cells of ideal network sources. If nets are specified, **size_only** is set on all cells that are cells of the specified nets' global driver pins. This guarantees that ideal network sources are not optimized away by compile.

NOTE: The implied **size_only** attribute set by this command overrides the four **FALSE** attribute values set by the **set_size_only** command. Use the **report_cell** command to determine if a cell is **size_only** or not.

DRC checking is turned off so the nets in an ideal network are free of **max_capacitance**, **max_fanout**, and **max_transition** design rule constraints. Disabling all of these features improves runtime and timing optimization; for example, by not resetting and scanning networks that you might want synthesized separately when using the **compile** command.

The latency and transition times of an ideal network are 0 by default, but you can override them by using the **set_ideal_latency** and **set_ideal_transition** commands.

If the **ideal_network** attribute is set on floating objects in a design, these objects are still optimized away during compile.

The **set_ideal_network** command will not prevent clock gating optimizations from modifying or removing clock gating cells. To prevent this use **set_dont_touch** command instead.

To reverse the effect of the **set_ideal_network** command, use the **remove_ideal_network** command and specify the source pins or ports for the network. Timing, source pins, and boundary pins of an ideal network are displayed by using the **report_ideal_network** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates an ideal network on objects in a design named *CLOCK_GEN*.

```
prompt> current_design CLOCK_GEN
prompt> set_ideal_network {port1 port2}
```

SEE ALSO

- compile(2)
- current_design(2)
- remove_ideal_network(2)
- report_cell(2)
- report_ideal_network(2)
- reset_design(2)
- set_auto_disable_drc_nets(2)
- set_dont_touch(2)
- set_dont_touch_network(2)
- set_ideal_latency(2)
- set_ideal_net(2)
- set_ideal_transition(2)
- set_size_only(2)

set_ideal_transition

Specifies ideal transition for the ideal network and ideal nets.

SYNTAX

```
status set_ideal_transition
[-rise | -fall]
[-min | -max]
transition_time
object_list
```

Data Types

```
transition_time  float
object_list      list
```

ARGUMENTS

-rise | -fall

Indicates whether the specified transition time is applicable for rising or falling transition. If you do not specify **-rise** or **-fall**, both values are set. The **-rise** and **-fall** options are mutually exclusive.

-min | -max

Specifies whether the transition is to be used for minimum delay analysis or maximum delay analysis. By default, the delay is used for minimum and maximum delay analysis. The **-min** and **-max** options are mutually exclusive.

transition_time

Specifies the ideal transition value on the pins in an ideal network. The transition time must be expressed in units consistent with the technology library used during optimization. For example, if the technology library specifies transition values in nanoseconds, *transition_time* must be expressed in nanoseconds.

object_list

Specifies a list of leaf cell pins or top-level ports that are the startpoints of the timing arcs for which ideal transition is set.

DESCRIPTION

Sets the ideal transition on top-level ports and leaf cell pins of an ideal network.

Design Compiler assumes ideal timing for the ideal network, which means pins have a specified ideal transition (from the **set_ideal_transition** command) or zero ideal transition by default. The ideal network is normally used for pre-layout to reduce runtime by avoiding unnecessary DRC optimization and retiming. Ideal transition provides an estimate of the ideal network for pre-layout.

The specified transition value overrides the internally-estimated cell and net transition value. If the specified pins do not belong to the ideal network, an error message is generated by the **report_ideal_network** command and *transition_time* is not used for those pins.

The ideal transition at an ideal boundary pin is the ideal transition of the closest pin with specified ideal transition.

You can use **set_ideal_transition** for pins at lower levels of the design hierarchy. Pins are specified in the form of "INSTANCE1/INSTANCE2/PIN_NAME."

To list ideal transition values, use the **report_ideal_network** command.

To remove the ideal transition values from a design, use the **remove_ideal_transition** or **reset_design** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example specifies a rise transition of 1.2 and a fall transition of 0.9 for the ports named A, B, and C.

```
prompt> set_ideal_transition 1.2 -rise {A B C}
prompt> set_ideal_transition 0.9 -fall {A B C}
```

SEE ALSO

- `remove_ideal_latency(2)`
- `remove_ideal_network(2)`
- `remove_ideal_transition(2)`
- `report_ideal_network(2)`
- `report_timing(2)`
- `set_ideal_latency(2)`
- `set_ideal_network(2)`

set_ieee_1500_configuration

Sets the IEEE 1500 insertion configuration for the current design.

SYNTAX

```
integer set_ieee_1500_configuration  
[-wir_width integer]
```

ARGUMENTS

-wir_width *integer*

Specifies the width of the IEEE 1500 wrapper instruction register (WIR) to be implemented by IEEE 1500 test-mode control insertion.

By default, the WIR width is automatically determined by the tool based on the number of test-mode encodings needed and the type of encoding specified by the **-mode_decoding_style** option of the **set_dft_configuration** command.

DESCRIPTION

This command specifies the configuration of the IEEE 1500 test-mode control logic to be inserted in the current design. Note that IEEE 1500 logic is implemented only when IEEE 1500 test-mode control is enabled by the **set_dft_configuration -ieee_1500 enable** command.

Use the **report_ieee_1500_configuration** command to display the current IEEE 1500 configuration of the design.

Use the **reset_ieee_1500_configuration** command to reset the current IEEE 1500 configuration of the design.

To configure IEEE 1500 control data registers, use the **set_scan_path** command.

EXAMPLES

The following command changes the WIR width to 4:

```
prompt> set_ieee_1500_configuration -wir_width 4
```

SEE ALSO

report_ieee_1500_configuration(2)
reset_ieee_1500_configuration(2)
set_scan_path(2)

set_ignored_layers

Specifies the routing layers that are ignored for RC estimation and congestion analysis. This command is supported only in topographical mode.

SYNTAX

```
status set_ignored_layers
  [-rc_congestion_ignored_layers layer_names]
  [-min_routing_layer min_name]
  [-max_routing_layer max_name]
```

Data Types

<i>layer_names</i>	list
<i>min_name</i>	string
<i>max_name</i>	string

ARGUMENTS

-rc_congestion_ignored_layers *layer_names*

Specifies the routing layers to be ignored during congestion analysis and RC estimation.

-min_routing_layer *min_name*

Specifies the minimum (lowest) layer that can be used for routing. The tool marks all routing layers below the specified layer as ignored layers for RC and congestion estimation.

-max_routing_layer *max_name*

Specifies the maximum (highest) layer that can be used for routing. The tool marks all routing layers above the specified layer as ignored layers for RC and congestion estimation.

DESCRIPTION

In Design Compiler topographical mode, this command sets the routing layers that are ignored during congestion analysis and RC estimation. You can use either the **-rc_congestion_ignored_layers** option to explicitly exclude specific layers from consideration, or the **-min_routing_layer** and **-max_routing_layer** options to specify the range of layers considered (causing all other layers to be ignored).

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the m5 and m6 layers to be ignored during congestion analysis and RC estimation:

```
prompt> set_ignored_layers -rc_congestion_ignored_layers {m5 m6}
```

The following example sets m2 as the minimum routing layer and m5 as the maximum routing layer. All layers below m2 and above m5 are set as ignored layers for congestion analysis and RC estimation.

```
prompt> set_ignored_layers -min_routing_layer m2 -max_routing_layer m5
```

SEE ALSO

```
remove_ignored_layers(2)  
report_ignored_layers(2)  
report_lib(2)
```

set_impl_priority

Sets the formula attribute of the priority parameter and/or the set_id attribute for implementations in synthetic libraries.

SYNTAX

```
status set_impl_priority
  [-priority formula]
  [-set_id id]
  implementation_list
```

Data Types

<i>formula</i>	string
<i>id</i>	int
<i>implementation_list</i>	list

ARGUMENTS

-priority *formula*

Specifies the string to which the **formula** attribute of the **priority** parameter should be set. An empty string causes the **priority** parameter to be removed from the specified implementations. The formula should evaluate to an integer between 0 and 10.

-set_id *id*

Specifies the value to which the **set_id** attribute should be set.

implementation_list

Specifies a list of implementations for which the **priority** parameter and/or the **set_id** attribute are to be set. Implementation names must contain a library prefix. For more information, refer to the EXAMPLES section of this man page. If more than one object is included, they must be enclosed in quotes or braces ({}).

DESCRIPTION

The **set_impl_priority** command sets the **formula** attribute of the **priority** parameter and/or the **set_id** attribute on the specified implementations. Either **-priority** or **-set_id** must be specified. The synthetic libraries containing the implementations must be loaded into **dc_shell**.

NOTE: This command cannot be used in **dc_shell** scripts embedded in HDL descriptions. In addition, the command affects only the copy of the library that is currently loaded into memory, and has no effect on the version that exists on disk. However, if the library is written out using **write_lib**, all **priority** parameters or **set_id** attributes will be written out, causing library objects to be permanently

changed.

EXAMPLES

The following command sets the formula attribute in the **priority** parameter to " $n < 4 ? 6 : 2$ " and the **set_id** attribute to 4 for 'my_lib.sldb/m1/rpl':

```
prompt> set_impl_priority -priority "n < 4 ? 6 : 2" -set_id 4 my_lib.sldb/m1/rpl
```

The following command sets the **set_id** attribute to 3 for all the implementations of module 'm2' in the synthetic library 'my_lib.sldb':

```
prompt> set_impl_priority -set_id 3 my_lib.sldb/m2/*
```

The following command removes the **priority** parameter for implementations 'imp1' and 'imp2' of the module 'syn_lib.sldb/mod':

```
prompt> set_impl_priority -priority "" { syn_lib.sldb/mod/imp1, syn_lib/mod/imp2 }
```

SEE ALSO

[compile\(2\)](#)

[report_synlib\(2\)](#)

set_implementation

Specifies the implementation to use for synthetic library cell instances in a design.

SYNTAX

```
status set_implementation  
    implementation_name  
    cell_list  
    [-check_impl]
```

Data Types

```
implementation_name    string  
cell_list              list
```

ARGUMENTS

implementation_name

Specifies the name of the implementation used to implement the function of the *cell_list* synthetic library cell instances. A single period (.) in the implementation name locks in the current implementation of the instance. The implementation name must also contain the module name, such as *DW01_add/cla*, when setting implementation on synthetic operators.

cell_list

Specifies the synthetic library cell instances to be implemented by *implementation_name*. If more than one instance name is specified, enclose the names in quotes (") or in braces ({}). Specify either instantiated cell instances or inferred cell instances, but not both, in *cell_list*.

-check_impl

Checks for the existence of the specified implementation before setting implementation on the cell instances.

DESCRIPTION

The **set_implementation** command tags specified synthetic library cell instances with the name of an implementation to use to implement them. When the **compile** command is run, the tool chooses the user-specified implementation to implement a cell instance. Otherwise, **compile** chooses the implementation it considers most appropriate for a cell instance. For a design that has already been compiled, **set_implementation** with a period (.) locks in the current implementation. Subsequent compiles do not change the implementation.

The **set_implementation** command does not function on cell instances that are not defined in a synthetic library.

The **report_resources** command lists the current implementation of all synthetic library instances, and indicates whether the implementation has been locked by **set_implementation**.

Use the **report_synlib** command for the possible implementations of a synthetic library cell.

EXAMPLES

If *A1* is an instance of the *DW01_ADD* cell in the current design, a *cla* carry-lookahead implementation can be specified to implement *A1*. The following example assumes that *DW01_ADD* is a module defined in a synthetic library listed in **synthetic_library**, and that *DW01_ADD* has a *cla* carry-lookahead implementation associated with it:

```
prompt> set_implementation cla A1
```

The following example directs **compile** to use *rpl* for both *Adder1* in the design *FIFO1* and *Adder2* in the design *FIFO2*:

```
prompt> set_implementation rpl { FIFO1/Adder1 FIFO2/Adder2 }
```

The following example removes any effects of **set_implementation** from all synthetic cells of the current design:

```
prompt> remove_attribute [get_cells *] implementation
```

The following example locks the implementation of all synthetic cells in the current design. On subsequent compiles, the implementation does not change.

```
prompt> set_implementation . ""
```

The following example sets a *cla* implementation on an ADD operator cell named *A1*:

```
prompt> set_implementation DW01_add/cla A1
```

SEE ALSO

- `compile(2)`
- `remove_attribute(2)`
- `report_resources(2)`
- `report_synlib(2)`
- `synthetic_library(3)`

set_input_delay

Sets input delay on pins or input ports relative to a clock signal.

SYNTAX

```
status set_input_delay
    delay_value
    [-reference_pin pin_port_name]
    [-clock clock_name]
    [-clock_fall]
    [-level_sensitive]
    [-network_latency_included]
    [-source_latency_included]
    [-rise]
    [-fall]
    [-max]
    [-min]
    [-add_delay]
    port_pin_list
```

Data Types

<i>delay_value</i>	float
<i>clock_name</i>	collection of 1 object
<i>port_pin_list</i>	collection

ARGUMENTS

delay_value

Specifies the path delay. The *delay_value* must be in units consistent with the technology library used during optimization. The *delay_value* represents the amount of time the signal is available after a clock edge. This represents a combinational path delay from the clock pin of a register.

-reference_pin *pin_port_name*

Specifies the clock pin or port to which the specified delay is related. If you use this option, and if propagated clocking is being used, the delay value is related to the arrival time at the specified reference pin, which is clock source latency plus its network latency from the clock source to this reference pin. The options *-network_latency_included* and *-source_latency_included* cannot be used at the same time as the *-reference_pin* option. For ideal clock network, only source latency is applied.

The pin specified with the *-reference_pin* option should be a leaf pin or port in a clock network, in the direct or transitive fanout of a clock source specified with the *-clock* option. If multiple clocks reach the port or pin where you are setting the input delay, and if the *-clock* option is not used, the command considers all of the clocks.

-clock *clock_name*

Specifies the clock to which the specified delay is related. If **-clock_fall** is used, **-clock *clock_name*** must be specified. If **-clock** is not specified, the delay is relative to time zero for combinational designs. For sequential designs, the delay is considered relative to a new clock with the period determined by considering the sequential cells in the transitive fanout of each port.

The *clock_name* can be either a string or collection of one object.

-clock_fall

Specifies that the delay is relative to the falling edge of the clock. The default is the rising edge.

-level_sensitive

Specifies that the source of the delay is a level-sensitive latch. This allows the tool to derive the setup and hold relationship for paths from this port as if the port is a level-sensitive latch. If **-level_sensitive** is not used, the input delay is treated as if it is a path from a flip-flop.

-network_latency_included

Specifies that the clock network latency is not added to the input delay value. If this option is not specified, the clock network latency of the related clock is added to the input delay value. It has no effect if the clock is propagated or the input delay is not specified with respect to any clock.

-source_latency_included

Specifies that the clock source latency is not added to the input delay value. If this option is not specified, the clock source latency of the related clock will be added to the input delay value. It has no effect if the input delay is not specified with respect to any clock.

-rise

Specifies that *delay_value* refers to a rising transition on specified ports of the current design. If neither **-rise** nor **-fall** is specified, rising and falling delays are assumed to be equal.

-fall

Specifies that *delay_value* refers to a falling transition on specified ports of the current design. If neither **-rise** nor **-fall** is specified, rising and falling delays are assumed equal.

-max

Specifies that *delay_value* refers to the longest path. If neither **-max** nor **-min** is specified, maximum and minimum input delays are assumed equal.

-min

Specifies that *delay_value* refers to the shortest path. If neither **-max** nor **-min** is specified, maximum and minimum input delays are assumed equal.

-add_delay

Specifies whether to add delay information to the existing input delay or to overwrite. The **-add_delay** option enables you to capture information about multiple paths leading to an input port that are relative to different clocks or clock edges.

For example, the following command removes all other maximum rise input delay from A, since **-add_delay** is not specified. Other input delay with a different clock or with **-clock_fall** is removed.

```
set_input_delay 5.0 -max -rise -clock phi1 {A}
```

In the following example, **-add_delay** is specified. If there is an input maximum rise delay for A relative to clock *phi1* rising edge, the larger value is used. The smaller value will not result in critical timing for maximum delay. For minimum delay, the smaller

value is used. If there is maximum rise input delay relative to a different clock or different edge of the same clock, it remains with the new delay.

```
prompt> set_input_delay 5.0 -max -rise -clock phi1 -add_delay {A}
```

port_pin_list

Specifies a list of input port or internal pin names in the current design to which *delay_value* is assigned. If more than one object is specified, the objects are enclosed in quotes (") or in braces ({}). If input delay is specified on a pin, the cell of the pin is set to size only to leave room for compile applying sizing on it.

DESCRIPTION

The **set_input_delay** command sets input path delay values for the current design. Used with **set_load** and **set_driving_cell**, the input and output delays characterize the operating environment of the current design.

A path starts from a primary input or clock of sequential element and ends at a sequential element or primary output. The **delay_value** to be specified is the delay between the startpoint and the object on which the **set_input_delay** is being set, relative to the clock edge.

The **set_input_delay** command sets input path delays on input ports relative to a clock edge. Input ports are assumed to have zero input delay, unless specified. For inout (bidirectional) ports, you can specify the path delays for both input and output modes.

To describe a path delay from a level-sensitive latch, use the **-level_sensitive** option. If the latch is positive-enabled, set the input delay relative to the rising clock edge; if it is negative-enabled, set the input delay relative to the falling clock edge. If time is being borrowed at that latch, add that time borrowed to the path delay from the latch when determining input delay.

The **characterize** command automatically sets input and output delay, drive, and load values based on the environment of a cell instance.

The timer adds input delay to path delay for paths starting at primary inputs and output delay for paths ending at primary outputs.

Use the **report_port** command to list input delays associated with ports.

To list input delays of internal pins, use **report_design**.

Use **remove_input_delay** or **reset_design** to remove input delay values.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example sets an input delay of 2.3 for ports IN1 and IN2 on a combinational design. Because the design is combinational, no clock is needed.

```
prompt> set_input_delay 2.3 {IN1 IN2}
```

The following example uses a clock collection and sets an input delay of 1.2 relative to the rising edge of CLK1 for all input ports in the design:

```
prompt> set_input_delay 1.2 -clock [get_clocks CLK1] [all_inputs]
```

The following example sets the input and output delays for the bidirectional port INOUT1. The input signal arrives at INOUT1 2.5 units after the falling edge of CLK1. The output signal is required at INOUT1 at 1.4 units before the rising edge of CLK2.

```
prompt> set_input_delay 2.5 -clock CLK1 -clock_fall {INOUT1}
prompt> set_output_delay 1.4 -clock CLK2 {INOUT1}
```

The following example has three paths to the IN1 input port. One of the paths is relative to the rising edge of CLK1. Another path is relative to the falling edge of CLK1. The third path is relative to the falling edge of CLK2. The **-add_delay** option is used to indicate that new input delay information will not cause old information to be removed.

```
prompt> set_input_delay 2.2 -max -clock CLK1 -add_delay {IN1}
prompt> set_input_delay 1.7 -max -clock CLK1 -clock_fall \
-add_delay {IN1}
prompt> set_input_delay 4.3 -max -clock CLK2 -clock_fall \
-add_delay {IN1}
```

In this example, two different maximum delays and two minimum delays for port A are specified using the **-add_delay** option. Because the information is relative to the same clock and clock edge, only the largest of the maximum values and the smallest of the minimum values are maintained, in this 5.0 and 1.1. If the **-add_delay** option is not used, the new information overwrites the old information.

```
prompt> set_input_delay 3.4 -max -clock CLK1 -add_delay {A}
prompt> set_input_delay 5.0 -max -clock CLK1 -add_delay {A}
prompt> set_input_delay 1.1 -min -clock CLK1 -add_delay {A}
prompt> set_input_delay 1.3 -min -clock CLK1 -add_delay {A}
```

SEE ALSO

- all_inputs(2)
- characterize(2)
- create_clock(2)
- remove_input_delay(2)
- report_design(2)
- report_port(2)
- reset_design(2)
- set_driving_cell(2)
- set_load(2)
- set_output_delay(2)

set_input_transition

Sets the **max_transition_rise**, **max_transition_fall**, **min_transition_rise**, or **min_transition_fall** attributes to the specified transition values on the specified input and inout ports.

SYNTAX

```
status set_input_transition  
      transition  
      [-rise] [-fall] [-min] [-max]  
      port_list
```

Data Types

transition float
port_list collection

ARGUMENTS

transition

Specifies a nonnegative transition value to which the **max_transition_rise**, **max_transition_fall**, **min_transition_rise** or **min_transition_fall** attributes are to be set on the ports in *port_list*. The *transition* is the transition time of a slope that drives the port, such that a higher transition value means longer delays. Thus, a *transition* of 0 is infinite transition, or no delay between the ports and all that is connected to them. *transition* must be in units consistent with the technology library used during optimization.

-rise -fall

Indicates that the **rise** or **fall** attributes of *port_list* are to be set to *transition*. If neither is specified, both are set to *transition*.

-min -max

Indicates that the transition value is to be applied for minimum or maximum delay analysis. If no value is specified for minimum analysis, the maximum value is used.

port_list

Specifies a list of names of input or inout ports in the current design, on which the transition value is to be set. If you specify more than one port, you must enclose the ports in either quotes or braces ({}).

DESCRIPTION

Sets the **max_transition_rise**, **max_transition_fall**, **min_transition_rise** or **min_transition_fall** attributes to *transition* on specified

input and inout ports in the current design.

Note: The **set_input_transition** command removes any corresponding rise or fall driving cell attributes and rise or fall drive attributes on the specified ports. If you want a fixed transition value, use **set_input_transition**, otherwise, use **set_driving_cell** instead of **set_drive**, if possible.

To view transition information on ports, use **report_port -verbose**. To remove transition attributes from ports, use **remove_attribute**. The **reset_design** command removes all attributes from a design, including the transition attributes.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example sets the maximum rise and fall transition values of ports A, B, and C to 7.0.

```
prompt> set_input_transition 7.0 {A B C}
```

The following example sets the maximum rise and fall transition values of all input ports to 7 and sets the maximum rise transition value on port B to 11.

```
prompt> set_input_transition 7 [all_inputs]
prompt> set_input_transition -rise 11 B
```

The following example sets the minimum fall transition value on port C to 13.

```
prompt> set_input_transition -fall -min 13.0 {C}
```

SEE ALSO

- all_inputs(2)
- remove_attribute(2)
- report_port(2)
- reset_design(2)
- set_drive(2)
- set_driving_cell(2)
- set_load(2)

set_isolate_ports

Specifies the ports to be isolated from the internal fanouts of the driver nets.

SYNTAX

```
int set_isolate_ports  
  [-type inverter | buffer]  
  [-driver cell_name]  
  [-force]  
  port_list
```

Data Types

```
cell_name  string  
port_list  list
```

ARGUMENTS

-type inverter | buffer

Specifies whether inverter pairs or buffers are used to isolate the ports. Note that Design Compiler might perform sizing optimization on the inserted cell, which can cause the actual isolation cell type to be different from the specified cell type. If this behavior is not desired, use the **-force** option with the **-type** option, to ensure that the specified type of isolation cell is inserted.

-driver *cell_name*

Indicates that the specified target library cell should be used to isolate the ports. The library cell can be either a buffer or an inverter. Note that Design Compiler might perform sizing optimization on the inserted cell, which can cause the actual driver cell to be different from the specified cell type. If this behavior is not desired, use the **-force** option with the **-driver** option, to ensure that the specified driver cell is used.

-force

Specifies that isolation must be performed on the ports, even if an isolation is not required because of the absence of internal loads on the nets driving the ports. Note: Isolation might not be performed if a buffer, inverter pair, or cell corresponding to the specified isolation logic, is already present.

port_list

Specifies the ports in the current design that are to be isolated.

DESCRIPTION

The **set_isolate_ports** command specifies the input or output ports in the current design that are to be isolated. For an output port, the **compile** command attempts to ensure that the net driving the port does not have any other internal fanout by inserting an isolation cell. For an input port, an isolation cell is inserted if the port drives multiple cells or if the port drives a pin of a cell that contains more than one input pin.

By default, no action is taken if output ports are already isolated from internal fanouts of their driver nets and input ports are already isolated from driving multiple cells. If the **-force** option is specified, isolation is performed on the ports even if the ports do not need isolation with above two conditions.

Note that bidirectional ports cannot be isolated using this command. In addition, no isolation is performed on a port by **compile** if the net connected to the port has a **dont_touch** attribute or if the port has been defined as a clock source.

Use the **remove_isolate_ports** or the **reset_design** command to remove a port from the list of ports to be isolated. Use the **report_isolate_ports** command to list the isolation status of these ports.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies that isolation is required on the output port qout with a buffer.

```
prompt> set_isolate_ports -type buffer qout
```

The following command specifies isolation using a particular cell from the target library on an output port.

```
prompt> set_isolate_ports -driver IVDAP qout
```

SEE ALSO

- compile(2)
- report_compile_options(2)
- remove_isolate_ports(2)
- report_isolate_ports(2)
- reset_design(2)

set_isolation

Sets the UPF strategy for inserting isolation cells between a specified power domain and other power domains.

SYNTAX

```
status set_isolation
  isolation_strategy_name
  -domain power_domain
  [-isolation_power_net isolation_power_net]
  [-isolation_ground_net isolation_ground_net]
  [-isolation_supply_set isolation_supply_set]
  [-isolation_supply isolation_supply_set]
  [-clamp_value 0 | 1 | latch]
  [-applies_to inputs | outputs | both]
  [-applies_to_boundary upper | lower | both]
  [-source source_supply_set_name]
  [-sink sink_supply_set_name]
  [-diff_supply_only true | false]
  [-elements objects]
  [-exclude_elements objects]
  [-no_isolation]
  [-force_isolation]
  [-name_prefix prefix_string]
  [-name_suffix suffix_string]
  [-isolation_signal isolation_signal]
  [-isolation_sense low | high]
  [-location self | parent | fanout]
  [-update]
```

Data Types

```
isolation_strategy_name  string
power_domain            string
isolation_power_net     string
isolation_ground_net    string
isolation_supply_set    string
source_supply_set_name  string
sink_supply_set_name    string
objects                  list
prefix_string           string
suffix_string           string
isolation_signal        string
```

ARGUMENTS

isolation_strategy_name

Specifies the new UPF isolation strategy name for the power domain.

-domain power_domain

Specifies the power domain name to which this UPF isolation strategy applies.

-isolation_power_net isolation_power_net

Specifies the power net used to power isolation cells inserted by the isolation strategy.

-isolation_ground_net isolation_ground_net

Specifies the ground net for isolation cells inserted by the isolation strategy.

-isolation_supply_set isolation_supply_set

Specifies the supply set whose power and ground functions supply power to isolation cells inserted by the isolation strategy. It is mutually exclusive with the **-isolation_power_net** and **-isolation_ground_net** options. This option is supported with UPF 2.0 standard.

-isolation_supply isolation_supply_set

Specifies the same functionality as **-isolation_supply_set**, except this option is supported with UPF 3.0 standard. Only one of **-isolation_supply_set** or **-isolation_supply** can be specified.

-clamp_value 0 | 1 | latch

Specifies the constant value of the isolation cell output when the isolation function is enabled: **0**, **1**, or **latch**. The default is **0**. A clamp value of **latch** is implemented by an isolation cell that latches the current data and holds that value constant during isolation.

-applies_to inputs | outputs | both

Specifies explicitly whether to apply the strategy to inputs only, to outputs only, or to both inputs and outputs of the power domain. The default is **outputs** unless the **-source** or **-sink** option is used, or the **-diff_supply_only** option is set to **true** and the supplies are defined by supply sets, in which case the default is **both**.

-applies_to_boundary upper | lower | both

Specifies explicitly whether to apply the strategy to upper boundaries(lowConns) only, to lower boundaries(highConns) only, or to both upper and lower boundaries of the power domain.

The default is

upper

- if the power domain does not have lower_domain_boundary attribute OR
- if the power domain lower_domain_boundary attribute set to FALSE

both

- if the power domain lower_domain_boundary attribute set to TRUE

-source source_supply_set_name

Restricts the strategy to only domain-crossing nets that are driven by a cell powered by the specified supply set.

-sink sink_supply_set_name

Restricts the strategy to only domain-crossing nets that fan out to one or more cells powered by the specified supply set.

-diff_supply_only true | false

When set to **true**, applies the strategy only to domain-crossing nets in which the driver and receivers are powered by different

supply sets or supply nets. The default is **false**, meaning no isolation restriction based on matching driver and receiver supplies. The **-source**, **-sink**, and **-diff_supply_only** options cannot be used at the same time.

-elements *objects*

Restricts the strategy to only the listed objects in this power domain. The objects can be design elements such as cells, pins, and ports. Cells should be in the **-elements** of the **create_power_domain** command. Pins and ports should be the interface of a power domain.

-exclude_elements *objects*

Excludes the listed objects from the strategy in this power domain. The objects can be design elements such as cells, pins, and ports.

-no_isolation

Prevents any isolation of elements specified by this strategy.

-force_isolation

Forces isolation of the elements specified for this strategy, irrespective of priority rules for conflicting strategies. Isolation cells inserted with this option cannot be optimized away by implementation tools.

-name_prefix *prefix_string*

Specifies the string to add to the beginning of names of isolation cells inserted by the strategy, overriding the **name_format** command. An empty string is allowed as a setting.

-name_suffix *suffix_string*

Specifies the string to add to the end of names of isolation cells inserted by the strategy, overriding the **name_format** command. An empty string is allowed as a setting.

-isolation_signal *isolation_signal*

Specifies the enable isolation signal for cells created by the isolation strategy. The *isolation_signal* can be a port, pin, or net.

-isolation_sense *low* | *high*

Specifies the sense of the enable isolation signal, either **low** or **high**. The default is **high**.

-location *self* | *parent* | *fanout*

Specifies the hierarchical location for inserting isolation cells created by the strategy: **self** (the default) to place isolation cells inside the design being isolated, **parent** to place isolation cells in the parent level of the design being isolated, or **fanout** to place the isolation cells at all fanout locations (sinks) of the port being isolated.

-update

Indicates that this command provides additional information for a previous command with the same *strategy_name* and *domain_name* and executed in the same scope. The additional information includes elements specified with the **-elements** or **-exclude_elements** options.

DESCRIPTION

The **set_isolation** command specifies an isolation strategy for a power domain. The strategy specifies the elements of the domain to which the strategy applies, the supply set or supply nets to be used for the isolation cells, and any insertion constraints based on the supply sets of the driver and receiver of the net being isolated. To complete the isolation strategy, you also need to use the

set_isolation_control command to specify the isolation control signal, and optionally, the hierarchical location for inserting the cells on the domain boundary.

Synthesis and implementation tools insert isolation cells strictly according the isolation strategies specified by the **set_isolation** and **set_isolation_control** commands, without considering other information such as the power state table. If the isolation strategies are not consistent with the allowed states defined in the power state table, you can find and report these inconsistencies by using the **check_mv_design** command after the isolation cells are inserted.

Where the Strategy Applies

By default, the strategy of a **set_isolation** command applies to all outputs that cross the boundary of the specified power domain.

To restrict the strategy to specific elements of the power domain, use the **-elements** option. The specified elements can be ports of the power domain or the pins of root cells on the boundary of the power domain. You can use the **set_isolation** command multiple times to create different isolation strategies for different elements in the power domain.

The ports of a power domain are the logical ports of the root cells of the power domain, or in the case of a power domain containing the top-level design, the logical ports of the design. Input ports of a power domain are the ports defined as inputs in the corresponding HDL module. Similarly, output ports of a power domain are the ports defined as outputs in the corresponding HDL module.

The **-exclude_elements** list explicitly identifies a set of ports to which this strategy does not apply. The *exclude_list* might contain instances or ports in the specified domain. If an instance name is specified in the *exclude_list*, it is equivalent to specifying all the ports of the instance in the *exclude_list*.

The **-update** option adds information to the base command executed in the same scope. When specified with the **-update** option, the **-elements** and **-exclude_elements** are additive. The **-update** option must be specified with the isolation name, the power domain name, and either the **-elements** or **-exclude_elements**. All other options are not allowed with the **-update** option.

The **-applies_to** option lets you specify explicitly whether to apply the strategy to **inputs** only, to **outputs** only, or to **both** inputs and outputs of the power domain:

- Applying the strategy to the outputs of a shutdown domain ensures that during shutdown, the outputs continue to drive the inputs of other domains with a known value (optionally specified by the **-clamp_value** option).
- Applying the strategy to the inputs of a power domain might be necessary if another domain drives the inputs, the other domain can be shut down, and the outputs of the other domain are not already isolated according to that domain's isolation strategy.

Without the **-applies_to** option, the strategy applies to both inputs and outputs of the domain if you use one or both of the **-source** and **-sink** options, or if you set the **-diff_supply_only** option to **true** and the supplies are defined by supply sets (rather than supply nets). Otherwise, the strategy applies to outputs only.

The tool, by strictly following the isolation strategies, might insert more isolation cells than necessary, for example, at both the output of the driver domain and the input of the receiver domain. You can find such occurrences with the **check_mv_design** command. The tool merges redundant cells during optimization.

Isolation Constraints Based on Source and Sink Supply Sets

The **-source**, **-sink**, and **-diff_supply_only** options restrict the insertion of isolation cells to only the domain-crossing nets whose driver and receiver cells meet specified supply set conditions. These options act as "filters" to reduce the number of ports on the domain boundary considered for isolation.

- The **-source** option specifies a supply set name. The isolation strategy applies only to the domain-crossing nets that are driven by a cell powered by the specified supply set.
- The **-sink** option specifies a supply set name. The isolation strategy applies only to the domain-crossing nets that fan out to cells powered by the specified supply set.

You can use both the **-source** and **-sink** options to restrict the insertion of isolation cells to only the domain-crossing nets that are driven by a cell powered by a specified supply set and that fan out to cells powered by a different specified supply set.

- The **-diff_supply_only** option, when set to **true**, applies the strategy only to the domain-crossing nets in which the driver and receivers are powered by different supply sets or supply nets. This setting prevents isolation when the driver and receiver cells use matching supplies. The default is **false**, meaning no restriction on isolation based on matching driver and receiver supplies.

Setting the **-diff_supply_only** option to **true** restricts the isolation strategy to all combinations of differing source and sink supply sets, whereas setting the **-source** and **-sink** options restrict the strategy to specifically named combinations of supply sets. Note that you cannot use the **-diff_supply_only**, **-source**, and **-sink** options all together in the same strategy.

Isolation Cell Supply Set or Supply Nets

The power and ground supplies for the inserted isolation cells must be active while the driver domain is shut down and the receiver domain is active. For each isolation strategy, the power supplies for the isolation cells must be specified by one of the following options:

- The **-isolation_supply_set** or **-isolation_supply** option of the **set_isolation** command
- The **-isolation_power_net** and **-isolation_ground_net** options of the **set_isolation** command
- The **-supply {default_isolation supply_set_name}** option of the **create_power_domain** command

Using the **-isolation_supply_set/-isolation_supply** option of the **set_isolation** command completely specifies the supplies for the isolation cells. Alternatively, you can specify the power and ground supply nets explicitly using one or both of the **-isolation_power_net** and **-isolation_ground_net** options. If you specify only the power net or only the ground net, the primary supply of the power domain is used for the unspecified supply net.

If you use none of these options in the **set_isolation** command, the **default_isolation** supply set specified by the **-supply** option of the **create_power_domain** command applies.

Clamp Value

The **-clamp_value** option specifies the constant value of the isolation cell output when the isolation function is enabled. It can be set to **0**, **1**, or **latch**. A clamp value of **0** (the default) can be implemented with an AND gate. A clamp value of **1** can be implemented with an OR gate. A clamp value of **latch** is implemented with an isolation cell that latches the current data, either 0 or 1, when the isolation function is enabled, and holds that value constant during isolation.

Additionally the clamp value that should be used when implementing an isolation strategy on a given port or pin can be set using the **-clamp_value** option of the **set_port_attributes** command. A **set_port_attribute -clamp_value** setting has precedence over the clamp value of the isolation strategy being implemented.

Forcing or Preventing Isolation

In case of conflicting isolation strategies, the tool follows certain precedence rules to determine which strategy to use. For details, see the *Power Compiler User Guide*.

Using the **-force_isolation** option of the **set_isolation** command ensures that an isolation strategy is applied to the specified elements, irrespective of the default priority order. The isolation cells inserted with this option cannot be optimized away by implementation tools.

To prevent the insertion of isolation cells for the specified elements of the power domain, use the **-no_isolation** option. In that case, there is no need to specify the isolation power supplies for the strategy, and there is no need for the **set_isolation_control** command for the strategy.

Isolation control signal, sense and location

The **-isolation_signal** argument specifies the enable isolation signal name. You can specify the enable signal as a port, pin, or net. A port or pin has priority over an identically named net.

The **-isolation_sense** option specifies the sense of the enable isolation signal, either **high** or **low**. The default is **high**.

The isolation signal need not exist in the logical hierarchy where the isolation cells are to be inserted; the synthesis or implementation tool can perform port-punching to make the connection. Port-punching means automatically creating a port to make a connection from one hierarchical level to the next.

Punched ports are not considered for isolation or level shifting. Existing ports, even if they reside on an always-on path, are isolated and level-shifted according to the applicable isolation and level shifter strategy.

The **-location** option defines where the isolation cells are placed in the logic hierarchy:

- The **self** setting (the default) places isolation cells just inside the power domain boundary.
- The **parent** setting places isolation cells in the parent level of the design being isolated, just outside the power domain boundary.
- The **fanout** setting places isolation cells at all fanout locations (sinks) of the port being isolated. This option is valid only for an isolation strategy defined by the **set_isolation** command using the **-source** and **-sink** options or the **-diff_supply_only true** setting.

Using the **fanout** setting might be necessary when an output port fans out to multiple power domains, each requiring a different type of isolation.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates an isolation strategy called ISO1, which applies all output ports of power domain PD1. The strategy specifies the power and ground nets used for the isolation cells inserted by the strategy.

```
prompt> set_isolation ISO1 -domain PD1 \
  -isolation_power_net PN1 -isolation_ground_net GN1
```

The following example creates an isolation strategy called ISO2, which applies only to port OUT2 of block instance U3 in power domain PD1. The power domain PD1 was defined on the block U3, a block that can be shut down.

```
prompt> set_isolation ISO2 -domain PD1 \
  -isolation_power_net PN2 -isolation_ground_net GN2 \
  -elements U3/OUT2
```

The following example creates an isolation strategy called ISO3, which applies only to port OUT3 of block instance U3 in power domain PD1. The strategy specifies the supply set used for isolation cells inserted by the strategy.

```
prompt> set_isolation ISO3 -domain PD1 \
  -isolation_supply_set MY_SS1 \
  -elements U3/OUT3
```

The following example creates an isolation strategy called ISO4, which applies only to port OUT4 of block instance U3 in power domain PD1. No supply set or supply net is specified in the command, so the tool uses the **default_isolation** supply set previously defined in the **-supply** option of the **create_power_domain** command (which created power domain PD1).

```
prompt> set_isolation ISO4 -domain PD1 \
        -elements U3/OUT4
```

The following example creates an isolation strategy using the **-source** and **-sink** options. The strategy applies only to domain-crossing nets that are driven by a cell powered by the PD_primary_set supply set and are received by one or more cells powered by the SSET supply set.

```
prompt> set_isolation ISO5 -domain PD1 \
        -isolation_supply_set MY_SS1 \
        -source PD_primary_set -sink SSET
```

The following example creates an isolation strategy using the **-source** and **-diff_supply_only** options. The strategy applies only to domain-crossing nets that are driven by a cell powered by the PD_primary_set supply set and are received by cells powered by a different supply set.

```
prompt> set_isolation ISO6 -domain PD1 \
        -isolation_supply_set my_iso_supply_set \
        -source PD_primary_set \
        -diff_supply_only true
```

The following example specifies the active low, isolation enable signal, ENISO. The strategy name is ISO1 in power domain PD1. Isolation cells are inserted at the parent level, just outside the power domain boundary.

```
prompt> set_isolation ISO1 -domain PD1 \
        -isolation_supply_set my_iso_supply_set \
        -isolation_signal ENISO -isolation_sense low \
        -location parent
```

The following example creates an isolation strategy using the **-name_prefix** and **-name_suffix** options. The name of the new isolation cell created for the domain boundary pin "m1/b1/OUT4" is "PD1_snps_PD1__ISO7_snps_OUT4_iso_1".

```
prompt> set_isolation ISO7 -domain PD1 \
        -isolation_supply_set my_iso_supply_set \
        -elements {m1/b1/OUT4} \
        -name_prefix PD1 -name_suffix iso_1
```

If you do not use the **-name_prefix** and **-name_suffix** options, the **name_format** command determines the prefix and suffix. If you do not use the **name_format** command either, the default suffix is "_UPF_ISO" and there is no prefix.

The following example specifies an isolation strategy that applies to lower domain boundary (highConn). In this example ISO1 applies to output pins which are highConns of power domain PD_TOP. Isolation cells will be inserted in power domain PD_TOP driving inputs of mid.

```
prompt> create_power_domain PD_TOP
prompt> create_power_domain PD_MID -elements {mid}
prompt> set_isolation ISO1 -domain PD_TOP \
        -applies_to_boundary lower
```

SEE ALSO

```
check_mv_design(2)
create_power_domain(2)
set_isolation_control(2)
name_format(2)
set_port_attributes(2)
```

set_design_attributes(2)

set_isolation_cell

Sets the specified library cells as isolation cells.

SYNTAX

```
status set_isolation_cell  
    cell_name  
    [-data_pin data_pin_name]  
    [-enable_pin enable_pin_name]
```

Data Types

```
cell_name      string  
data_pin_name  string  
enable_pin_name string
```

ARGUMENTS

cell_name

Specifies the name of the library cell to be defined as an isolation cell.

-data_pin *data_pin_name*

Specifies the name of the data pin of the isolation cell.

-enable_pin *enable_pin_name*

Specifies the name of the enable pin of the isolation cell.

DESCRIPTION

The **set_isolation_cell** command sets the specified library cells as isolation cells. You can also specify the data and enable pin details for the isolation cell.

EXAMPLES

In the following example, all cells in the library whose name starts with ISO are set as isolation cells. The EN pin of those library cells

is considered the enable pin of the isolation cell.

prompt> **set_isolation_cell ISO* -enable_pin EN**

SEE ALSO

report_lib(2)
check_library(2)

set_isolation_control

Specifies the isolation control signal and the side of the domain boundary on which to insert isolation cells.

SYNTAX

```
status set_isolation_control  
  isolation_strategy  
  -domain power_domain  
  -isolation_signal isolation_signal  
  [-isolation_sense low | high]  
  [-location self | parent | fanout]
```

Data Types

<i>isolation_strategy</i>	string
<i>power_domain</i>	string
<i>isolation_signal</i>	string

ARGUMENTS

isolation_strategy

Specifies the name of a UPF isolation strategy previously created by the **set_isolation** command. The option settings apply to this strategy.

-domain *power_domain*

Specifies the name of the power domain previously created by the **create_power_domain** command to which the strategy options apply.

-isolation_signal *isolation_signal*

Specifies the name of the signal that enables and disables the isolation function of cells created by the isolation strategy. The *isolation_signal* can be a port, pin, or net.

-isolation_sense low | high

Specifies the sense of the isolation signal, either **low** or **high**, that enables the isolation function of cells created by the strategy. The default is **high**.

-location self | parent | fanout

Specifies the hierarchical location for inserting isolation cells created by the strategy: **self** (the default) to place isolation cells inside the design being isolated, **parent** to place isolation cells in the parent of the design being isolated, or **fanout** to place the isolation cells at all fanout locations (sinks) of the port being isolated.

DESCRIPTION

This command specifies the isolation control signal for a specified isolation strategy, and optionally, the hierarchical location for inserting isolation cells on the domain boundary.

The **-isolation_signal** argument specifies the name of the control signal that enables and disables the isolation function of cells created by the isolation strategy. You can specify control signal as a port, pin, or net. A port or pin has priority over an identically named net.

The **-isolation_sense** option specifies the logic state of the isolation control signal, either **high** or **low**, that enables the isolation function of the isolation cells. The default is **high**.

The isolation signal need not exist in the logical hierarchy where the isolation cells are to be inserted; the synthesis or implementation tool can perform port-punching as needed to make the connection. Port-punching means automatically creating a port to make a connection from one hierarchical level to the next.

Punched ports are not considered for isolation or level shifting, even though these ports might reside within the coverage of an isolation or level shifter strategy. Existing ports, even if they reside on an always-on path, are isolated and level-shifted according to the applicable isolation and level shifter strategy.

The **-location** option defines where the isolation cells are placed in the logic hierarchy:

- The **self** setting (the default) places isolation cells just inside the power domain boundary.
- The **parent** setting places isolation cells in the parent of the design being isolated, just outside the power domain boundary.
- The **fanout** setting places isolation cells at all fanout locations (sinks) of the port being isolated. This option is valid only for an isolation strategy defined by the **set_isolation** command using the **-source** and **-sink** options or the **-diff_supply_only true** setting.

Using the **fanout** setting might be necessary when an output port fans out to multiple power domains, each requiring a different type of isolation.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies the control signal, ENISO, that enables the isolation function for isolation cells inserted by the strategy named ISO1 in power domain PD1. A logic low on the signal enables the isolation function. The command also specifies that isolation cells are to be inserted in at the parent level, just outside the power domain boundary.

```
prompt> set_isolation_control ISO1 -domain PD1 \  
-isolation_signal ENISO -isolation_sense low \  
-location parent
```

SEE ALSO


```
create_power_domain(2)  
set_isolation(2)
```

set_keepout_margin

Creates a keepout margin of the specified type for the specified cell or library cell.

SYNTAX

```
status set_keepout_margin
  [-type hard | soft]
  [-outer {lx by rx ty}]
  [-tracks_per_macro_pin value]
  [-min_padding_per_macro value]
  [-max_padding_per_macro value]
  [-all_macros]
  [-macro_masters]
  [-macro_instances]
  [-north]
  [object_list]
```

Data Types

<i>lx</i>	float
<i>by</i>	float
<i>rx</i>	float
<i>ty</i>	float
<i>value</i>	float
<i>object_list</i>	float

ARGUMENTS

-type hard | soft

Specifies the type of the keepout margin to be created. The value can be either **soft** or **hard**. By default, the type is **hard**.

-outer {*lx by rx ty*}

Specifies the margin parameters for a keepout. The values **lx**, **by**, **rx** and **ty** represent the left, bottom, right, and top margins respectively. The margin parameters must be positive and all four parameters are required. The units are microns. If the margins are set on a standard cell instance or a standard cell master, and the standard cell instance is fixed, then all four margins are honored. However, if the standard cell instance is movable, only **lx** and **rx** are honored for horizontal designs and only **by** and **ty** are honored for vertical designs.

Setting a top or bottom keepout margin on movable standard cells is not recommended. If a standard cell has a top or bottom keepout margin, the top or bottom keepout margin is usually ignored. If the movable standard cell becomes a fixed cell, then cell overlaps might be reported when you run the **check_legality** command. For example, clock tree synthesis commands, such as **compile_clock_tree** and **optimize_clock_tree**, might change movable standard cells to fixed standard cells. As a result, cell overlaps might be reported by the **check_legality** command. To avoid this problem, do not set a top or bottom keepout margin on

movable standard cells. Use the top or bottom keepout margin only on fixed standard cells.

-tracks_per_macro_pin value

Specifies the number of tracks per macro pin. This value is used to calculate the keepout margin around macro cells based on pin count. A pin count-based keepout margin type is always **hard**, the **-type** option is ignored when you specify the **-tracks_per_macro_pin** option. The **-all_macros**, **-macro_masters**, and **-macro_instances** options are not allowed together with the **-tracks_per_macro_pin** option. By default, the value is 0.5 tracks per pin.

-min_padding_per_macro value

Specifies the minimum padding per macro size. This value is used to calculate the keepout margin based on pin count. The **-type** option is ignored. The **-all_macros**, **-macro_masters** and **-macro_instances** options are not allowed when you specify the **-min_padding_per_macro** option. Note: the margin calculation for the left and right macro sides considers the *height* of the cell, the top and bottom sides consider the *width* of the cell. By default, the value is 0, which is the minimum grid size.

-max_padding_per_macro value

Specifies the maximum padding per macro. This value is used to calculate the keepout margin based on pin count. The **-type** option is ignored. The **-all_macros**, **-macro_masters** and **-macro_instances** options are not allowed when you specify the **-max_padding_per_macro** option. Note: the margin calculation for the left and right macro sides considers the *height* of the cell, the top and bottom sides consider the *width* of the cell. By default, the value is -1.

-all_macros

Specifies that the keepout margin of the specified type is created on all macro cells in the design.

-macro_masters

Creates keepout margins of the specified type on all macro masters or the macro masters of the macro instances specified in the *object_list*. Note: If you specify this option, the keepout margin is created on only macro masters, the tool ignores all standard cells and standard library cells specified in the *object_list*.

-macro_instances

Creates keepout margins of the specified type on all macro instances or all the macro instances of the macro masters specified in the *object_list*. If you specify macro master instances in the object list, the tool creates the keepout margin on macro instances of the macro master. Note: If you specify this option, the keepout margin is created on only the macro instances, the tool ignores all standard cells and standard library cells specified in the *object_list*.

-north

Specifies that the specified keepout values are with respect to the north orientation of the cell.

object_list

Specifies the list of cells or library cells for which to create keepout margins.

DESCRIPTION

The **set_keepout_margin** command creates keepout margins for the specified cells and library cells. The tool derives the keepout margins for the cells in the following order.

1. Keepout margin specified on the cell itself
2. Keepout margin specified on the library cells of the cell
3. Pin count-based derived keepout margin values for macro cells

Note that the tool derives pin count-based keepout margin values only for the macro cells. If the tool does not find the keepout margin on the cell, it looks for the keepout margin specified on the library cells. For macro cells, if there is no user-specified keepout margin on cells and lib cells, the tool looks for the pin count-based derived keepout margin values. A keepout is created for a cell or library cells whether or not the cell has a `fixed_placement` attribute. Keepout margins can be set on ILM or block abstraction cells and references also.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets a keepout margin on MY_CELL and MY_LIB_CELL.

```
prompt> set_keepout_margin -type hard \  
-outer {10 10 10 10} MY_CELL  
prompt> set_keepout_margin -type hard \  
-outer {10 10 10 10} MY_LIB_CELL
```

The following command sets a keepout margin on the ILM reference blk.

```
prompt> set_keepout_margin -type hard \  
-outer {10 10 10 10} [get_ilms -reference blk]
```

The following command sets a keepout margin on all the macro cells in the design.

```
prompt> set_keepout_margin -type hard -all_macros \  
-outer { 10 10 10 10}
```

This sets a keepout margin on the basis of pin count of the macro.

```
prompt> set_keepout_margin -tracks_per_macro_pin .6 \  
-min_padding_per_macro .1 -max_padding_per_macro 0.2
```

The following command sets a keepout margin on all the ILM instances in the design.

```
prompt> set_keepout_margin -type hard \  
-outer {10 10 10 10} [get_ilms]
```

The following command sets a keepout margin on all the block abstraction instances in the design.

```
prompt> set_keepout_margin -type hard \  
-outer {10 10 10 10} [get_cells -hierarchical -filter "@is_block_abstraction==true"]
```

SEE ALSO

`get_attribute(2)`
`remove_keepout_margin(2)`
`report_keepout_margin(2)`

set_latch_loop_breakers

Specifies transparent latch data pins to be used as loop-breaker latch data pins when the variable **timing_enable_through_paths** is set to true.

SYNTAX

```
string set_latch_loop_breakers  
  -pin pin_list  
  [-remove]  
  [-avoid]
```

Data Types

pin_list list

ARGUMENTS

-pin *pin_list*

Specifies data pins of transparent latches that are to be made loop-breakers (when no other command options are used).

-remove

Removes previously specified settings on the listed pins.

-avoid

Requests that the tool to avoid choosing the listed pins when it selects loop-breaker pins. These requests cannot always be honored, especially if there is a loop from the pin through combinational logic to itself.

DESCRIPTION

When sequential loops of transparent latches exist in a design, the tool selects certain latch data pins for special analysis as loop-breaker data pins. Reporting paths through these latch data pins is not allowed except by using the *-trace_latch_borrow* feature of the **report_timing** command.

The **set_latch_loop_breakers** command allows you to guide the tool in selecting the pins to be used as loop-breaker data pins. By guiding the tool to select some latch data pins and not others, you are more likely to be able to report the paths of interest.

This command has no effect if the variable **timing_enable_through_paths** is not enabled.

EXAMPLES

The following example specifies that the pin L1/D should be selected as a loop-breaker pin.

```
prompt> set_latch_loop_breakers -pin [get_pin L1/D]
```

SEE ALSO

report_latch_loop_groups(2)
get_latch_loop_groups(2)
timing_enable_through_paths(3)

set_leakage_optimization

Enables or disables leakage-power optimization on the current design. **Note:** This command is no longer needed for the **compile_ultra** command because leakage-power optimization is enabled by default and cannot be disabled within the **compile_ultra** command. This man page describes the previous usage for this command.

SYNTAX

status **set_leakage_optimization**
true | false

ARGUMENTS

true | false

Controls whether leakage-power optimization is enabled.

When the setting is **true**, enables leakage-power optimization. When the setting is **false**, disables leakage-power optimization.

DESCRIPTION

This command enables or disables leakage-power optimization on the current design. This command does not support scenario-specific settings in a multicorner-multimode design. To perform scenario-dependent leakage-power optimization, use the **set_scenario_option -leakage_power true** command.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example enables leakage-power optimization in a non-multicorner-multimode design.

```
prompt> set_leakage_optimization true
1
```

The following example disables leakage-power optimization in a non-multicorner-multimode design.

```
prompt> set_leakage_optimization false
```

1

SEE ALSO

set_dynamic_optimization(2)
set_scenario_options(2)

set_leakage_power_model

Specifies the model that will be optimized by leakage optimizations.

SYNTAX

```
status set_leakage_power_model
      [-type model_name]
      [-mvth_weights weights]
      [-reset]
```

Data Types

```
model_name  string
weights     string
```

ARGUMENTS

-type *model_name*

Specifies the name of the leakage power model. Valid model names are **leakage** and **channel_width**. The default model is **leakage**.

-mvth_weights *weights*

Specifies the design-level weights for the voltage groups. The library level values specified in the technology library are ignored and the values specified with the **-mvth_weights** option are used to calculate the weighted sum of channel widths.

-reset

Resets the model to **leakage** and clears the weights that were previously set.

DESCRIPTION

Specifies the model that will be optimized by leakage. The leakage optimizations during compile optimize a model. One of the following models may be chosen as the leakage power model:

```
leakage
channel_width
```

When the model is **leakage**, the leakage power is computed and optimized. When the model is **channel_width**, a weighted sum of channel width is computed and optimized.

The default model is **leakage**.

For this command to work, the library must contain:

- threshold-voltage groups
- the channel width for cells

Furthermore, if the `threshold_voltage_channel_width_factor` is not present in the library, you must assign `-mvth_weights` at the design level.

EXAMPLES

In this example, the weights specified at the library level are used to calculate the weighted sum:

```
prompt> set_max_leakage_power 0
prompt> set_leakage_power_model -type channel_width
prompt> compile_ultra
```

The weights specified in the library are used to calculate the weighted sum. The following example illustrates the specification of design-level weights. This specification overrides the library-level weights specified in the technology library.

```
prompt> set_max_leakage_power
prompt> set_leakage_power_model -type channel_width \
    -mvth_weights "lvt = 1000 nvt = 300 hvt = 1"
prompt> compile_ultra
```

SEE ALSO

`report_power(2)`
`set_max_leakage_power(2)`

set_level_shifter

Sets a strategy for level shifting during implementation.

SYNTAX

```
status set_level_shifter
  level_shifter_name
  -domain domain_name
  [-elements list]
  [-exclude_elements list]
  [-applies_to inputs | outputs | both]
  [-applies_to_boundary upper | lower | both]
  [-threshold value]
  [-rule low_to_high | high_to_low | both]
  [-location self | parent | automatic]
  [-no_shift]
  [-force_shift]
  [-source source_supply_set_name]
  [-sink sink_supply_set_name]
  [-name_prefix prefix]
  [-name_suffix suffix]
  [-update]
```

Data Types

<i>level_shifter_name</i>	string
<i>domain_name</i>	string
<i>list</i>	list
<i>value</i>	float
<i>source_supply_set_name</i>	string
<i>sink_supply_set_name</i>	string
<i>prefix</i>	string
<i>suffix</i>	string

ARGUMENTS

- level_shifter_name***
- Specifies the level-shifter strategy name, which is used only for reporting. This is a required argument.
- domain *domain_name***
- Specifies the domain for which the strategy is applied. This is a required argument.
- elements *list***

Specifies a list of design elements, pins, or ports to which this strategy is applied.

-exclude_elements *list*

Specifies a list of design elements, pins, or ports to which this strategy is not applied.

-applies_to inputs | outputs | both

Specifies whether the domain's input ports, output ports, or both are level shifted. The default is **both**.

-applies_to_boundary upper | lower | both

Specifies explicitly whether to apply the strategy to upper boundaries(lowConns) only, to lower boundaries(highConns) only, or to both upper and lower boundaries of the power domain.

The default is

upper

- if the power domain does not have lower_domain_boundary attribute OR
- if the power domain lower_domain_boundary attribute set to FALSE

both

- if the power domain lower_domain_boundary attribute set to TRUE

-threshold *value*

Specifies the voltage threshold (in volts) for determining when level-shifter cells are required. The default is **0**.

-rule low_to_high | high_to_low | both

Specifies the type of level-shifter cells that are required. The default is **both**.

-location self | parent | automatic

Specifies where the level-shifter cell is placed in the logic hierarchy. The default is **automatic**.

-no_shift

Prevents the insertion of level-shifter cells on the specified ports, pins, and nets when used with the **-elements** option.

-force_shift

Allows specific and unconditional level-shifter insertion if possible.

-source *source_supply_set_name*

Causes level shifter insertion to occur only on ports where the source matches the provided supply set.

-sink *sink_supply_set_name*

Causes level shifter insertion to occur only on ports where the sink matches the provided supply set.

-name_prefix *prefix*

Specifies the prefix string for the names of newly inserted level-shifter cells. Prefix can also be an empty string. This option overrides the settings specified by the **name_format** command.

-name_suffix *suffix*

Specifies the suffix string for the names of newly inserted level-shifter cells. Suffix can also be an empty string. This option overrides the settings specified by the **name_format** command.

-update

Specifies extra elements to be added to the **-elements** and **-exclude_elements** lists of a level shifter strategy, which must exist

previously.

DESCRIPTION

The **set_level_shifter** command is used to set a strategy for level shifting during implementation. Level-shifter cells are placed on the signals that have sources and sinks operating at different voltages, as their associated design elements are connected to different supply nets.

If a level-shifter strategy is not specified on a power domain, the default level-shifter strategy consists of all elements in the power domain, and uses the default strategy settings. Power state tables and operating conditions are considered to determine if level-shifter cells are needed for design elements on the boundaries of power domains.

If **-elements** *list* is specified, the elements must be in the domain specified by **-domain** *domain_name*.

If **-exclude_elements** *list* is specified, the strategy doesn't apply to the elements on the list, whether they belong to the boundary of specified domain or are explicitly included using the **-elements** option. If an excluded element didn't previously apply to the strategy, then its presence on the list is ignored.

The **-threshold** option defines how large the voltage difference between the driver and the sink needs to be before level-shifter cells are inserted. Normally, this threshold value is determined from the cell libraries. Use the **-threshold** option to override the library values.

The **-rule** value can be **low_to_high**, **high_to_low**, or **both**.

- If **low_to_high** is specified, signals going from a lower voltage to a higher voltage get a level-shifter cell when the voltage difference exceeds that specified by **-threshold**.
- If **high_to_low** is specified, signals going from a higher voltage to a lower voltage get a level-shifter cell when the voltage difference exceeds that specified by **-threshold**.
- If **both** is specified, it is equivalent to having specified both rules in the strategy.

The **-location** option defines where the level-shifter cells are placed in the logic hierarchy. All necessary supplies need to be available in the specified location. The option values are as follows:

- **self** specifies that the level-shifter cell is placed inside the model or cell being shifted.
- **parent** specifies that the level-shifter cell is placed in the parent of the cell or model being shifted.
- **automatic** specifies that the implementation tool chooses the appropriate locations.

The **-no_shift** option cannot be specified with the **-rule**, **-location** or **-threshold** options. In this case, only the **-no_shift** option is used.

The **-force_shift** option cannot be specified with the **-no_shift** option. If **-force_shift** option is used with a **-threshold** option, the threshold is ignored as if it is not specified. If the **-force_shift** option is used along with a **-rule** option, the rule determines the type of **-force_shift** level shifter inserted. The **-location** option might be used along with the **-force_shift** option to specify the location where the level shifter is required.

The **-update** option adds design elements to the **-elements** and **-exclude_elements** of an existing level shifter strategy, as if they had been specified with the original command. Adding design elements to the **-elements** list is allowed only if the first **set_level_shifter** command which defined the strategy also included the **-elements** option. In other words, a domain-based strategy cannot be changed to an elements-based strategy. The *level_shifter_name*, *strategy name*, and the *domain_name*, and at least one of the **-elements** or **-exclude_elements**, are required options when specifying the **-update** option. No other option is allowed.

Note that this command does not apply to inout ports. Also, it is an error if the specified location is not within the logic design starting

at the design root.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the level-shifter strategy on PowerDomainZ:

```
prompt> set_level_shifter shift_up -domain PowerDomainZ -applies_to outputs \
        -threshold 0.02 -rule both
```

The following example shows how **-name_prefix** and **-name_suffix** are used and how they are specified for a UPF level-shifter strategy.

The name of the new level-shifter cell that is created for the domain boundary pin "m1/b1/x" is "PD1_x_ls_1" in this example.

```
prompt> set_level_shifter ls_1 -domain PD1 \
        -elements {m1/b1/x} \
        -location self \
        -name_prefix PD1_ \
        -name_suffix _ls_1
```

If you have not specified the **-name_prefix** and **-name_suffix** options, the **-level_shift_prefix** and **-level_shift_suffix** options specified by the **name_format** command are used as the prefix and suffix substrings for the level-shifter cell.

In the following example, the name of the new level-shifter cell that is created for the domain boundary pin "m1/b1/x" is "top_x_ls".

```
prompt> set_level_shifter ls_1 -domain PD1 \
        -elements {m1/b1/x} \
        -location self
prompt> name_format -level_shift_prefix "top_" \
        -level_shift_suffix "_ls"
```

If you have not specified the **-level_shift_prefix** and **-level_shift_suffix** options specified by the **name_format** command either, the level-shifter cell name is suffixed with the "_UPF_LS" substring. By default, the level-shifter cell name is not prefixed with any substring.

In the following example, the name of the new level-shifter cell that is created for the domain boundary pin "m1/b1/x" is "x_UPF_LS".

```
prompt> set_level_shifter ls_1 -domain PD1 \
        -elements {m1/b1/x} \
        -location self
```

The following example shows how the UPF level-shifter strategy and isolation strategy options **-name_prefix** and **-name_suffix** are used to name the inferred enable level-shifter cell.

In the following example, the name of the new enable level-shifter cell that is created for the domain boundary pin "m1/b1/x" is "PD1_x_iso_1_ls_1".

```
prompt> set_isolation iso_1 -domain PD1 \
        -isolation_supply_set iso_supply_set \
        -elements {m1/b1/x} \
        -name_prefix PD1_ \
        -name_suffix _iso_1
```

```
prompt> set_level_shifter ls_1 -domain PD1 \
  -elements {m1/b1/x} \
  -location self \
  -name_prefix PD1_ \
  -name_suffix _ls_1
```

In this example, the **-name_prefix** that is specified with the **set_isolation** and **set_level_shifter** commands is the same, so the prefix substring is used only when naming the inferred enable level-shifter cell.

If you have not specified the **-name_prefix** and **-name_suffix** options for the **set_isolation** or **set_level_shifter** commands, the **-isolation_prefix**, **-isolation_suffix**, **-level_shift_prefix**, and **-level_shift_suffix** options specified by the **name_format** command are used to decide the prefix and suffix substrings for the enable level-shifter cell.

In the following example, the name of the new enable level-shifter cell that is created for the domain boundary pin "m1/b1/x" is "top_ls_top_iso_x_iso_ls_1".

```
prompt> set_isolation iso_1 -domain PD1 \
  -isolation_supply_set iso_supply_set \
  -elements {m1/b1/x}
prompt> set_level_shifter ls_1 -domain PD1 \
  -elements {m1/b1/x} \
  -location self \
  -name_suffix _ls_1
prompt> name_format -level_shift_prefix "top_ls_" \
  -level_shift_suffix "_ls_" \
  -isolation_prefix "top_iso_" \
  -isolation_suffix "_iso"
```

If you have not specified the **-level_shift_prefix**, **-level_shift_suffix**, **-isolation_prefix**, or **-isolation_suffix** options specified by the **name_format** command either, the enable level-shifter cell name is suffixed with the "_UPF_ISO" substring. By default, the enable level-shifter cell name is not prefixed with any substring.

In the following example, the name of new enable level-shifter cell that is created for the domain boundary pin "m1/b1/x" is "x_UPF_ISO".

```
prompt> set_isolation iso_1 -domain PD1 \
  -isolation_supply_set iso_supply_set \
  -elements {m1/b1/x}
prompt> set_level_shifter ls_1 -domain PD1 \
  -elements {m1/b1/x} \
  -location self
```

In the following example applies level-shifter strategy ls_1 on all output pins of block m1, except its m1/in1 pin. Strategy is not applied to any pin of block m2.

```
prompt> set_level_shifter ls_1 -domain PD1 \
  -location self
  -elements {m1 m2/in1 m2/in2}
prompt> set_level_shifter ls_1 -domain PD1 \
  -exclude_elements {m1/in1 m2}
  -update
```

The following example specifies a level-shifter strategy that applies to lower domain boundary (highConn). In this example LS1 applies to output pins which are highConns of power domain PD_TOP. Level-shifter cells will be inserted in power domain PD_TOP driving inputs of mid.

```
prompt> create_power_domain PD_TOP
prompt> create_power_domain PD_MID -elements {mid}
prompt> set_level_shifter LS1 -domain PD_TOP \
```

-applies_to_boundary lower -applies_to outputs

SEE ALSO

add_port_state(2)
add_pst_state(2)
create_pst(2)
report_pst(2)
name_format(2)

set_level_shifter_cell

Sets on-the-fly specification of a library level-shifter cell.

SYNTAX

```
status set_level_shifter_cell  
  cell_name  
  [-cell_type cell_type]  
  [-cell_input_voltage_range {lower_range upper_range}]  
  [-cell_output_voltage_range {lower_range upper_range}]  
  [-std_cell_main_rail_pg_pin pg_pin_name]  
  [-data_pin data_pin_name]  
  [-input_voltage_range {lower_range upper_range}]  
  [-output_voltage_range {lower_range upper_range}]  
  [-input_signal_level signal_level]  
  [-enable_pin enable_pin_name]  
  [-enable_signal_level signal_level]  
  [-output_signal_level signal_level]
```

Data Types

<i>cell_name</i>	string
<i>lower_range</i>	float
<i>upper_range</i>	float
<i>pg_pin_name</i>	string
<i>data_pin_name</i>	string
<i>signal_level</i>	string
<i>enable_pin_name</i>	string

ARGUMENTS

cell_name

Specify the name of the level-shifter library cell.

-cell_type *cell_type*

Specifies the cell type of the level-shifter cell.

-cell_input_voltage_range {*lower_range* *upper_range*}

Specifies the cell input voltage range of the level-shifter cell.

-cell_output_voltage_range {*lower_range* *upper_range*}

Specifies the cell output voltage range of the level-shifter cell.

-std_cell_main_rail_pg_pin *pg_pin_name*

Specifies the standard cell P/G pin of the level-shifter cell.

-data_pin *data_pin_name*

Specifies the data pin of the level-shifter cell.

-input_voltage_range {*lower_range upper_range*}

Specifies the input voltage range of the level-shifter cell.

-output_voltage_range {*lower_range upper_range*}

Specifies the output voltage range of the level-shifter cell.

-input_signal_level *signal_level*

Specifies the input signal level of the level-shifter cell.

-enable_pin *enable_pin_name*

Specifies the enable pin of the level-shifter cell.

-enable_signal_level *signal_level*

Specifies the enable signal level of the level-shifter cell.

-output_signal_level *signal_level*

Specifies the output signal level of the level-shifter cell.

DESCRIPTION

The **set_level_shifter_cell** command specifies the properties of the level-shifter cells.

EXAMPLES

In the following example, library cells whose names start with LVLLH are set as level-shifter cells of type low to high.

```
prompt> set_level_shifter_cell LVLLH* -cell_type LH -std_cell_main_rail_pg_pin VDD \
-enable_pin EN -input_signal_level VDDL -enable_signal_level VDDH \
-output_signal_level VDDH -cell_input_voltage_range {0.7 1.4} \
-cell_output_voltage_range {0.7 1.4}
```

SEE ALSO

report_lib(2)
check_library(2)

set_lib_attribute

Sets the value of an attribute on a library object.

SYNTAX

```
list set_lib_attribute  
  object_list  
  attribute_name  
  attribute_value
```

Data Types

<i>object_list</i>	list
<i>attribute_name</i>	string
<i>attribute_value</i>	datatype of attribute_name

ARGUMENTS

object_list

A list of the library objects on which the attribute is to be set.

attribute_name

The name of the attribute to set.

attribute_value

The value of the attribute. The datatype must be the same as that of the attribute.

DESCRIPTION

Sets the value of an attribute on a library object. Only the attribute values of a small set of library objects can be set using this command. For the complete list of such library objects and their related "updatable" attributes, please refer to the user manual.

This command returns the list of objects that have the specified attribute value set. A returned empty list means that the command was not successful.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This command is used to find the cell "INV" and set its area value to 1.0:

```
prompt> set_lib_attribute sample/INV area 1.0  
Performing set_lib_attribute on library object 'sample/INV'.  
{sample/INV}
```

In this example, an error is issued because the cell "INV1" is not found:

```
prompt> set_attribute sample/INV1 area 1.0  
Error: The 'INV1' object does not exist in the 'sample' technology library. (UIL-54)  
{}
```

In the following command, the "lib_udg_attr" value for the user-defined group "lib_udg1" in the library "sample" is set to "test":

```
prompt> set_lib_attribute sample/lib_udg1 lib_udg_attr test  
Performing set_lib_attribute on library object 'sample/lib_udg1'.  
{sample/lib_udg1}
```

In the following command, the area values for gates "G1" and "G2" in the library "tech_lib" are both set to 2.0:

```
prompt> get_lib_attribute {sample/G1, sample/G2} area  
Performing set_lib_attribute on library object 'sample/G1'.  
Performing set_lib_attribute on library object 'sample/G2'.  
{sample/G1 sample/G2}
```

SEE ALSO

`get_attribute(2)`

set_libcell_dimensions

Sets the width and height of a library cell.

SYNTAX

```
int set_libcell_dimensions  
  -cell cell_name  
  -width width  
  -height height
```

Data Types

```
cell_name  string  
width      float  
height     float
```

ARGUMENTS

-cell *cell_name*

Specifies the name of the cell in the target library.

-width *width*

Specifies the width (dimension in x direction) of the library cell.

-height *height*

Specifies the height (dimension in y direction) of the library cell.

DESCRIPTION

The **set_libcell_dimensions** command sets the width and height of the specified library cell. All the arguments to this command are required. The units of width and height are in microns.

During **reoptimize_design**, LBO will use the pin location in conjunction with the cell location, dimension, and orientation to determine the absolute location of the pin. This will improve the accuracy of delay prediction during LBO, especially when large cells such as RAMs and large macros are used.

EXAMPLES

In the following example, the width of library cell INVERT_2 is set to 10 microns and the height is set to 15 microns.

```
prompt> set_libcell_dimensions -cell INVERT_2 -width 10 -height 15
```

SEE ALSO

`set_libpin_location(2)`

set_libcell_subset

Restricts the optimization of sequential cells and instantiated combinational cells to a family of target libraries.

SYNTAX

```
status set_libcell_subset  
-object_list cells  
-family_name name
```

Data Types

```
cells    list  
name     string
```

ARGUMENTS

-object_list *cells*

Specifies a list of sequential cells or instantiated combinational cells for which the target library family is used during optimization. The cells must be either unmapped or instantiated to one of the library cells in the target library family.

-family_name *name*

Specifies the target library family to be used for optimization of these cells. A family is a subset of target library cells and is defined by the **define_libcell_subset** command.

DESCRIPTION

The **set_libcell_subset** command specifies a family of target libraries that are available for use during the optimization of the specified instances. A family is defined by the **define_libcell_subset** command.

The target library family restriction applies only to the new cells that are created or mapped during optimization. This restriction does not affect the cells that are already mapped. Therefore, the unoptimized portions of the design are not affected. Use the **report_libcell_subset** command to see the cells that are already mapped to the library cells outside their subset.

This command is supported only in topographical mode.

To remove the target library family from an instance, use the **remove_libcell_subset** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the target library family for the u1 and u2 instances to special_flops, which consists of the library cells, SDFLOP1 and SDFLOP2.

```
prompt> define_libcell_subset -libcell_list "SDFLOP1 SDFLOP2" -family_name special_flops  
prompt> set_libcell_subset -object_list [get_cells u1 u2] -family_name special_flops
```

SEE ALSO

```
remove_libcell_subset(2)  
report_libcell_subset(2)  
set_libcell_subset(2)  
define_libcell_subset(2)
```

set_libpin_location

Sets the location of a pin of a library cell relative to the origin of the library cell.

SYNTAX

```
int set_libpin_location  
-cell library_cell_name  
-pin pin_name_of_the_library_cell  
-coordinate {x_coordinate y_coordinate}
```

Data Types

```
library_cell_name      string  
pin_name_of_the_library_cell  string
```

ARGUMENTS

-cell *library_cell_name*

Specifies the name of the library cell.

-pin *pin_name_of_the_library_cell*

Specifies the name of the pin of the library cell.

-coordinate {*x_coordinate* *y_coordinate*}

Specifies the x and y coordinates of the pin. These coordinates are relative to the origin of the cell specified. The units of the coordinates are in microns.

DESCRIPTION

The **set_libpin_location** command sets the location of the specified pin of the specified library cell. All the arguments to this command are required.

This command should be issued before **reoptimize_design**. During **reoptimize_design**, LBO will use the pin location in conjunction with the cell location and orientation to determine the absolute location of the pin. This will improve the accuracy of delay prediction during LBO, especially when large cells such as RAMs and large macros are used.

Specify **set_libpin_location** only once for all the pins on the library cell, irrespective of the number of times the cell is instantiated or where it will be instantiated.

EXAMPLES

In the following example, location of the pin Z of library cell INVERT_2 is set to {10.1 5.2}.

```
prompt> read_clusters design.pdef prompt> set_libpin_location -cell INVERT_2 -pin A -coordinate {10.1 5.2}
```

SEE ALSO

set_port_location(2)

set_link_library_subset

Restricts the selection of library cells so they are chosen from a subset of the libraries specified by the **link_library** variable. This command can resolve ambiguity among libraries with the same voltage, temperature, and process.

SYNTAX

```
status set_link_library_subset
  [-object_list cells]
  [-top]
  [library_list]
```

Data Types

```
cells      list
library_list list
```

ARGUMENTS

-object_list *cells*

Specifies a list of cells for which the link library subset is used. The cells must be instances of hierarchical designs, macros, or pads. The subset restriction applies to these and their child instances. You must specify either the **-top** option or the **-object_list** option. Otherwise, the link library subset is not applied.

-top

Sets the link library subset on the current design. You must specify either the **-top** option or the **-object_list** option. Otherwise, the link library subset is not applied.

library_list

Specifies a subset of libraries belonging to the **link_library** variable. For any reference name that appears in a library of the subset, only library cells within the subset are used. For reference names that do not appear in the subset, library cells are selected from any library in the **link_library** variable, as usual. If *library_list* is not specified, the subset is empty at the specified hierarchy, and therefore all characterizations are selected from the full **link_library**.

DESCRIPTION

The **set_link_library_subset** command restricts the set of libraries from which library cells will be selected. You can use the command to resolve ambiguity among libraries that are characterized for the same voltage, temperature, and process, but that are characterized differently for other known arbitrary parameters.

In a multivoltage or multivoltage design, the tool would normally determine a cell's timing and power from a library in the **link_library** whose voltage, temperature, and process match the cell. If more than one library satisfied these conditions, an "Ambiguous Libraries" warning (MV-086) would be reported. By using the **set_link_library_subset** command, you can identify which library is appropriate for a particular block and/or scenario.

The command does not override existing library selection rules; it augments them. The tool still matches voltage, temperature, and process within the libraries listed in the subset. The link library subset provides an additional filter on the libraries to be considered.

Reference names that are not included in the subset libraries are not affected by the subset. Their characterizations are selected from all the libraries in the link library, as usual. You could, for example, use the command to make macro libraries unambiguous by specifying just the relevant macro library, without the need to list all the other libraries in the link library. See the Examples section.

Macros created by the PrimeTime **extract_model** command, for different modes, might still need to be merged with the PrimeTime **merge_models** command. A LIBSETUP-030 error message is issued if a macro's related library cells have different timing arcs. For example, the same cell across multiple libraries must have the same characteristics, such as **when** conditions and number of pins. Only the timing and power numbers can vary. If these conditions are not met, the tool issues an error.

To remove a link library subset from the design or a design instance, use the **remove_link_library_subset** or **reset_design** command.

As with most design constraints, the subset specifications are tied to the current design. If you change the current design to another level, and want the subset to affect optimization at that level, you need to reapply the **set_link_library_subset** command, or use the **characterize** or **propagate_constraints** command.

Multicorner-Multimode Support

This command affects the current scenario only. The same block can have different link library subsets in different scenarios. Because the command is scenario-specific, it must be specified in each scenario where it should take effect.

EXAMPLES

The following example has two libraries, lib1.db and lib2.db, with the same voltage, temperature, and process. Ordinarily this would result in an "Ambiguous Libraries" warning, but it does not because the ambiguity is resolved. Library values are taken from lib1.db everywhere in the design, except for the hierarchy under cell u1, which uses values from lib2.db.

```
prompt> set link_library "*" lib1.db lib2.db"
prompt> set_link_library_subset "lib1.db" -top
prompt> set_link_library_subset "lib2.db" -object_list [get_cells u1]
```

The following example uses two macro libraries, extracted as models by PrimeTime with different parasitics in effect.

```
prompt> set link_library "*" stdlib.db macro_corner1.db macro_corner2.db"
prompt> create_scenario A
prompt> set_link_library_subset "macro_corner1.db" -object_list [get_cells u1/my_macro]
prompt> create_scenario B
prompt> set_link_library_subset "macro_corner2.db" -object_list [get_cells u1/my_macro]
```

This example has the same effect as the previous example. The libraries in the subsets only contain the macro part name, so only this name is affected by the subsets. Any instance of this macro name will use the subset, but all other cells, such as standard cells, will be taken from the full link library, as usual.

```
prompt> set link_library "*" stdlib.db macro_corner1.db macro_corner2.db"
prompt> create_scenario A
prompt> set_link_library_subset "macro_corner1.db" -top
prompt> create_scenario B
```

```
prompt> set_link_library_subset "macro_corner2.db" -top
```

SEE ALSO

remove_link_library_subset(2)
report_link_library_subset(2)
link_library(3)

set_load

Sets the **load** attribute on the specified ports and nets.

SYNTAX

```
status set_load  
  value  
  objects  
  [-subtract_pin_load]  
  [-min]  
  [-max]  
  [[-pin_load] [-wire_load]]
```

Data Types

value float
objects list

ARGUMENTS

value

Specifies the value to which to set the **load** attribute on the ports and nets specified in the *objects* argument. The *value* argument must be expressed in units consistent with the technology library used during optimization. For example, if the technology library specifies load values in picofarads, the *value* argument must also be expressed in picofarads.

objects

Specifies a list of ports and nets in the current design whose loads are to be set.

-subtract_pin_load

Indicates that the current pin capacitances of the net are to be subtracted from the specified value before the net load value is set. If the resulting net load value is negative, it is set to 0.

This option sets the **subtract_pin_load** attribute on the specified nets. When this attribute is set, the total load computed on these nets during the **update_timing** command does not include pin loads. Note that the **subtract_pin_load** attribute is not placed on ports.

Use this option if the *value* argument includes pin and port capacitances.

-min

Indicates that the load value is to be used for minimum delay analysis.

If no minimum load value is specified, the maximum value is used.

-max

Indicates that the load value is to be used for maximum delay analysis.

If no value is specified for maximum analysis on a net, and a value has been specified for minimum analysis, the minimum value is ignored. (You cannot annotate only a minimum value on a net.)

-pin_load -wire_load

Indicates whether the specified load value on the port is to be treated as a pin load, a wire load, or both.

These options can be used only with ports; an error message is displayed if the *objects* argument contains any nets. If you do not specify either **-pin_load** or **-wire_load**, **-pin_load** is used as the default. You can use both arguments together, in which case the load value is set as both a pin load and as a wire load on the specified ports.

DESCRIPTION

This command sets the **load** attribute on the specified ports and nets in the current design.

If the current design is hierarchical, it must have been linked with the **link** command.

The total load on a net is the sum of all of pin loads, port loads, and wire loads associated with that net. The specified load value overrides the internally-estimated net load value.

You also can use the **set_load** command for nets at lower levels of the design hierarchy. These nets are specified as BLOCK1/BLOCK2/NET_NAME.

If you use the **-wire_load** option, the load value is set as a wire load on the specified port. However, the value is actually counted as part of the total wire load and not as part of the pin or port load.

To view load values on ports, use the **report_port** command. To view load values on nets, use the **report_net** or **report_internal_loads** command.

To reset a load value, use the **remove_attribute** command. To reset all annotated load values in a design, use the **reset_design** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example sets a load of 2 units on the port named in1:

```
prompt> set_load 2 in1
```

The following example sets a load of 2.5 units (the estimated wire load) plus the load of 3 inverter input pins from the tech_lib library on all output ports. The user-defined **port_load** variable is used to store this load value.

```
prompt> link -all
prompt> set port_load [expr 2.5 + 3 * [load_of tech_lib/IV/A]]
prompt> set_load $port_load [all_outputs]
```


The following example sets a load of pin Z of IV from the tech_lib library to the input_1 and input_2 ports using the **load_of** command:

```
prompt> set_load [load_of tech_lib/IV/Z] {input_1 input_2}
```

In the following example, a load of 3 is set on the U1/U2/NET3 net. The wire capacitance is set to 3. (Total net capacitance is 3 plus the sum of the pin and port capacitances.)

```
prompt> set_load 3 U1/U2/NET3
```

In the following example, a total net capacitance (wire capacitance plus pin capacitances) of 3 is set on the U1/U2/NET3 net. If the pin and port capacitances equal 2, the wire capacitance is annotated with 1. If the pin and port capacitances are 3 or more, the wire capacitance is annotated with 0.

```
prompt> set_load -subtract_pin_load 3 U1/U2/NET3
```

The following example sets a wire load of 5 units on the port named in1:

```
prompt> set_load -wire_load 5 in1
```

The following commands remove the back-annotated load on a port and on a net:

```
prompt> remove_attribute [get_ports in1] load
```

```
prompt> remove_attribute [get_nets U1/U2/NET3] load
```

SEE ALSO

- all_outputs(2)
- link(2)
- load_of(2)
- remove_attribute(2)
- report_internal_loads(2)
- report_net(2)
- report_port(2)
- reset_design(2)
- set_drive(2)
- set_wire_load_min_block_size(2)
- set_wire_load_mode(2)
- set_wire_load_model(2)
- set_wire_load_selection_group(2)
- target_library(3)

set_local_link_library

Sets the **local_link_library** attribute to specified files and libraries on the current design.

SYNTAX

```
int set_local_link_library  
    local_link_library
```

Data Types

local_link_library string

ARGUMENTS

local_link_library

Specifies a list of files with which the **local_link_library** attribute is to be set on the current design. The **local_link_library** files are not loaded or checked until the **link_library** is used.

DESCRIPTION

Sets the **local_link_library** attribute to *local_link_library* on the current design. The *local_link_library* is a list of design files and libraries that is added to the beginning of the **link_library** whenever a link operation is performed. Files in the *local_link_library* are searched first.

The **compile**, **insert_dft**, and **translate** commands set this attribute to the same value as the **target_library** variable.

Use **report_design** or **get_attribute** to identify the **local_link_library** for a design.

To remove this attribute, use the **remove_attribute** or **reset_design** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command sets the **local_link_library** "tech_lib.db" on the current design "my_design":

```
set_local_link_library
```

```
prompt> set_local_link_library tech_lib.db
```

Setting local link library 'tech_lib.db' on design 'my_design'

SEE ALSO

- compile(2)
- current_design(2)
- insert_dft(2)
- link(2)
- remove_attribute(2)
- reset_design(2)
- translate(2)
- design_attributes(3)
- link_library(3)
- target_library(3)

set_logic_dc

Specifies one or more input ports in the current design that are to be driven by don't care. The **set_logic_one** and **set_logic_zero** commands are used the same way as this command.

SYNTAX

```
int set_logic_dc  
  port_list
```

Data Types

port_list list

ARGUMENTS

port_list

A list of input port names in the current design that are to be driven by don't care. You can use only one of **set_logic_one**, **set_logic_zero**, or **set_logic_dc** on any given port. If you specify more than one port, you must include them in braces ({}).

DESCRIPTION

Assigns a **driven_by_dont-care** attribute to the ports in *port_list*; a given port can have only one of the **driven_by_logic_one**, **driven_by_logic_zero**, and **driven_by_dont-care** attributes. This information is used by **compile** to create smaller designs by eliminating logic that is tied to a specific value and therefore might not need to be maintained during optimization. After optimization, a port connected to logic one, logic zero, or don't care usually does not drive anything inside the optimized design.

Note:

This command cannot be used on output ports. To specify output ports as unconnected, use the **set_unconnected** command.

When a **set_logic_dc** command is used on an input port, **compile** is given the freedom to assign any signal to that input (including, but not limited to, zero and one). The meaning of this command is that the outputs of the design are significant only when the non-don't care inputs completely determine all outputs, independent of the don't care inputs. This information is used to optimize the design.

For example, suppose the design is a 2:1 multiplexer with inputs S, A, B, and output Z. The function computed is as follows:

$$Z = S * A + S * B$$

If you issue the following command,

```
prompt> set_logic_dc B
```

the implication is that the value of Z is significant only when S=1, and is a don't care when S=0. The resulting simplification done by **compile** represents the reduced logic as a wire, and the function is as follows:

$$Z = A$$

Use the **remove_attribute** command to remove these attributes.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following commands are issued to drive input ports "A" and "B" by logic 0, input port "C" by logic 1, and input port "D" by don't care:

```
prompt> set_logic_zero {A B}
```

```
prompt> set_logic_one C
```

```
prompt> set_logic_dc D
```

SEE ALSO

- get_attribute(2)
- remove_attribute(2)
- reset_design(2)
- set_logic_one(2)
- set_logic_zero(2)
- set_unconnected(2)
- simplify_constants(2)

set_logic_one

Specifies one or more input ports in the current design that are to be driven by logic one. The **set_logic_zero** and **set_logic_dc** commands are used the same way as this command.

SYNTAX

```
int set_logic_one  
  port_list
```

Data Types

port_list list

ARGUMENTS

port_list

A list of input port names in the current design that are to be driven by logic one. You can use only one of **set_logic_one**, **set_logic_zero**, or **set_logic_dc** on any given port. If you specify more than one port, you must include them in braces ({}).

DESCRIPTION

Assigns a **driven_by_logic_one** attribute to the ports in *port_list*; a given port can have only one of the **driven_by_logic_one**, **driven_by_logic_zero**, and **driven_by_dont-care** attributes. This information is used by **compile** to create smaller designs by eliminating logic that is tied to a specific value and therefore might not need to be maintained during optimization. After optimization, a port connected to logic one, logic zero, or don't care usually does not drive anything inside the optimized design.

Note:

This command cannot be used on output ports. To specify output ports as unconnected, use the **set_unconnected** command.

Use the **remove_attribute** command to remove these attributes.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following commands are issued to drive input ports "A" and "B" by logic 0, input port "C" by logic 1, and input port "D" by don't care:

```
prompt> set_logic_zero {A B}
```

```
prompt> set_logic_one C
```

```
prompt> set_logic_dc D
```

SEE ALSO

- `get_attribute(2)`
- `remove_attribute(2)`
- `reset_design(2)`
- `set_logic_dc(2)`
- `set_logic_zero(2)`
- `set_unconnected(2)`
- `simplify_constants(2)`

set_logic_zero

Specifies one or more input ports in the current design that are to be driven by logic zero. The **set_logic_one** and **set_logic_dc** commands are used the same way as this command.

SYNTAX

```
int set_logic_zero  
    port_list
```

Data Types

port_list list

ARGUMENTS

port_list

A list of input port names in the current design that are to be driven by logic zero. You can use only one of **set_logic_one**, **set_logic_zero**, or **set_logic_dc** on any given port. If you specify more than one port, you must include them in braces ({}).

DESCRIPTION

Assigns a **driven_by_logic_zero** attribute to the ports in *port_list*; a given port can have only one of the **driven_by_logic_one**, **driven_by_logic_zero**, and **driven_by_dont-care** attributes. This information is used by **compile** to create smaller designs by eliminating logic that is tied to a specific value and therefore might not need to be maintained during optimization. After optimization, a port connected to logic one, logic zero, or don't care usually does not drive anything inside the optimized design.

Note:

This command cannot be used on output ports. To specify output ports as unconnected, use the **set_unconnected** command.

Use the **remove_attribute** command to remove these attributes.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following commands are issued to drive input ports "A" and "B" by logic 0, input port "C" by logic 1, and input port "D" by don't care:

```
prompt> set_logic_zero {A B}
```

```
prompt> set_logic_one C
```

```
prompt> set_logic_dc D
```

SEE ALSO

- `get_attribute(2)`
- `remove_attribute(2)`
- `reset_design(2)`
- `set_logic_dc(2)`
- `set_logic_one(2)`
- `set_unconnected(2)`
- `simplify_constants(2)`

set_logicbist_configuration

Specifies the LogicBIST compression configuration for the design.

SYNTAX

```
status set_logicbist_configuration
  [-chain_count chain_count
  | -max_length max_chain_length]
  [-clock clock_name]
  [-pattern_counter_width register_width]
  [-shift_counter_width register_width]
  [-prpg_width register_width]
  [-misr_width register_width]
  [-burn_in disable | enable]
  [-occ_clock_weights weight_list]
  [-reset_weights weight_list]
  [-power_ramp_up disable | enable]
  [-power_ramp_down disable | enable]
  [-base_mode base_mode_name]
  [-test_mode logicbist_mode_name]
  [-self_test_mode autonomous_mode_name]
  [-num_shift_power_groups Number_of_shift_power_groups]
  [-shift_power_groups true | false]
  [-shift_power_ratio number_of_chains_per_shift_group]
```

Data Types

<i>chain_count</i>	integer
<i>max_chain_length</i>	integer
<i>clock_name</i>	string
<i>register_width</i>	integer
<i>weight_list</i>	list
<i>base_mode_name</i>	string
<i>logicbist_mode_name</i>	string
<i>autonomous_mode_name</i>	string

ARGUMENTS

-chain_count *chain_count*

Specifies the number of scan chains to build. The tool attempts to meet this constraint. If unable to do so, the tool issues a warning. See the DESCRIPTION section for more information.

-max_length *max_chain_length*

Specifies the maximum allowed length of the compressed scan chains. The tool attempts to meet this constraint. If unable to do so, the tool issues a warning. See the DESCRIPTION section for more information.

-clock *clock_name*

Specifies the scan clock to be used for the LogicBIST codec (compressor and decompressor) to be inserted. The specified clock must be previously declared as a test clock using the **set_dft_signal** command. By default, the tool selects the first-defined scan clock.

-pattern_counter_width *register_width*

Specifies the width of the pattern counter register. The maximum number of patterns is determined by the width of this register as follows:

$$\text{max\_patterns} == (2 \wedge \text{pattern\_counter\_register\_width}) - 2$$

The default is 8, which allows a maximum of 254 patterns.

-shift_counter_width *register_width*

Specifies the width of the shift counter register, which counts down the shift cycles for each pattern.

By default, the tool sizes this register according to the longest shift chain in the design. You normally do not need to specify this value.

-prpg_width *register_width*

Specifies the width of the pseudo-random pattern generator (PRPG) register. This register generates pattern data that feeds the compressed scan chains through an XOR phase shifter. The PRPG seed value is also this width.

By default, the tool sizes this register using a heuristic based on the number of compressed scan chains.

-misr_width *register_width*

Specifies the width of the multiple-input signature register (MISR). This register captures data from the compressed scan chains through an XOR compactor. The final MISR signature value is also this width.

If this register width is set to the same value as the number of compressed scan chains, the tool does not implement an XOR compactor.

By default, the tool sizes register this using a heuristic based on the number of compressed scan chains.

-burn_in disable | enable

Specifies whether to enable burn-in capability.

By default, self-test always stops with a pass/fail status after running self-test once. When burn-in capability is implemented, self-test can be configured to repeat continuously.

The burn-in behavior is configured by two signals, **lbistBurnInEnable** and **lbistBurnInStopOnFail**. See their descriptions in the **set_dft_signal** man page.

-occ_clock_weights *weight_list*

Specifies a list of clock signal groups and their weights.

By default, all OCC clocks capture in each pattern. If asynchronous cross-clock paths exist, you can use this option to group clocks into capturing groups. Each capture group has a weight value that specifies its probability of capture relative to the other groups.

Each group is itself a list containing an integer weight value followed by one or more clock signals:

`-occ_clock_weights {{75 SYSCLK} {25 CLK33 CLK66}}`

-reset_weights *weight_list*

Specifies a list of reset signal groups and their weights.

This option is similar to the **-occ_clock_weights** option, except that it is used for asynchronous reset signals.

-power_ramp_up disable | enable

Specifies whether to run power ramp-up patterns prior to self-test.

By default, when self-test starts, it immediately begins operating at the provided clock frequencies. If the design was previously in an idle/reset state, this could cause a transient drop in the power rail voltage.

When this option is set to **enable**, the self-test logic is augmented to shift through two "ramp-up" dummy patterns - one at 1/4 frequency, then one at 1/2 frequency - before beginning full-frequency self-test. The ramp-up patterns perform no testing and are not included in the pattern count; the seed value is loaded in the PRPG after ramp-up completes.

-power_ramp_down disable | enable

Specifies whether to run power ramp-down patterns after self-test completes.

This is similar to the **-power_ramp_up** option, except that the logic performs two "ramp-down" dummy patterns - one at 1/2 frequency, then one at 1/4 frequency - after self-test completes. The status signals are not asserted until the ramp-down patterns complete.

The ramp-down feature requires that the ramp-up feature also be enabled.

-base_mode *base_mode_name*

Specifies the standard scan base mode that is to be associated with the LogicBIST mode indicated by the **-test_mode** option. The default is **Internal_scan**.

-test_mode *logicbist_mode_name*

Specifies the LogicBIST test mode to which the specification applies. The default is *all_dft*, except when **define_test_mode** is used. In this case, the default is the value of the last *test_mode* defined.

Note that this is the test mode used by TetraMAX for seed/signature computation; it is not the test mode used for autonomous self-test operation.

-self_test_mode *autonomous_mode_name*

Specifies the test mode that must be in place during autonomous self-test operation. The LBIST_EN signal enables autonomous self-test only when this test mode's encoding is asserted on the test-mode signals. This mode cannot be a scan compression mode of any type (LogicBIST, DFTMAX, DFTMAX Ultra, etc.). The default is the mission-mode test mode.

Note that this is the test mode used for autonomous self-test operation; it is not the test mode used by TetraMAX for seed/signature computation.

DESCRIPTION

The **set_logicbist_configuration** command allows you to specify certain LogicBIST insertion parameters prior to using the **insert_dft** command. The parameters include specifications such as compressed scan chain count, pattern count register width, and MISR and PRPG register widths.

There are two related options to control the number of compressed scan chains for LogicBIST insertion, **-chain_count** and -

max_length. You must specify one of these options to configure LogicBIST compression. If both are specified, the **-max_length** option takes precedence over the **-chain_count** option.

SEE ALSO

define_test_mode(2)
insert_dft(2)
preview_dft(2)
report_logicbist_configuration(2)
reset_logicbist_configuration(2)

set_map_only

Sets the **map_only** attribute on specified objects so that they can be excluded from logic-level optimization during **compile**.

SYNTAX

```
status set_map_only  
  object_list  
  [flag]
```

Data Types

```
object_list  list  
flag         string
```

ARGUMENTS

object_list

Specifies a list of cells, references, or designs on which to set the **map_only** attribute.

flag

Determines the value to which the **map_only** attribute is to be set. Valid values are true (the default) or false.

DESCRIPTION

The **set_map_only** command places the **map_only** attribute on the specified objects, and sets the attribute value to either true or false. The default value of the **map_only** attribute is false. Setting the value to true excludes specified objects from logic-level optimization (flattening and structuring) during **compile**. Setting the **map_only** attribute to false resets the attribute to its default value, allowing the objects to be included in logic-level optimization.

To remove the **map_only** attribute, use the **remove_attribute** command. Alternatively, you can set **map_only** to false, which has the same effect as removing it. The **reset_design** command removes all attributes from a design, including **map_only**.

When the **map_only** attribute is set to true on a cell, reference, or design, the optimization of that part is treated specially. In general, **compile** attempts to map the cell exactly in the target library. For example, if you flag a GTECH_ADD_ABC cell as **map_only**, **compile** will try to find a full adder in your target library to use in the implementation. This allows you to suggest to **compile** (in a technology-independent manner) to use a gate that it otherwise might not have used.

While **map_only** suggests a gate for **compile** to use, it does not guarantee the gate's use. If **compile** cannot find a gate in your target library that has the correct function, **map_only** will be ignored. If the **map_only** gate is fed by a constant, **compile** will ignore the **map_only** attribute in order to simplify the gate.

The **map_only** attribute may be overridden in order to meet timing constraints. When **compile -map_effort** is set to high, an extra synthesis is done on the design. This synthesis selectively replaces **map_only** gates in order to meet design constraints. Gates that are **map_only** are kept if they are not on the critical path or have proved to be the best (fastest) choice for optimization. The **map_only** gates are remapped only if they are preventing constraints from being met.

EXAMPLES

In the following example, the **map_only** attribute is set to true on the U1 cell:

```
prompt> set_map_only U1
```

If the **map_only** attribute is set on a reference object, cells using that reference have the **map_only** attribute set during **compile**. In the following example, the **map_only** attribute is set to false on the U1 cell:

```
prompt> set_map_only U1 false
```

In the following example, all instances of MUX21 in the current design are considered **map_only**:

```
prompt> set_map_only [get_references MUX21]
```

If the **map_only** attribute is set on a library design object, cell instances of that design have the **map_only** attribute during **compile**. In this example, all the instances of AND2 from **my_library** are considered **map_only**:

```
prompt> set_map_only my_library/AND2
```

SEE ALSO

- `compile(2)`
- `current_design(2)`
- `get_references(2)`
- `remove_attribute(2)`
- `reset_design(2)`

set_max_area

Sets the **max_area** attribute to a specified value on the current design.

This command is only supported in DC Expert with the **compile** command. If you specify a **max_area** value and run the **compile_ultra** command, the **max_area** value is ignored and the default (zero) is used.

SYNTAX

```
int set_max_area  
  [-ignore_tns]  
  area_value
```

Data Types

area_value float

ARGUMENTS

-ignore_tns

Specifies that the area is prioritized above total negative slack (TNS).

area_value

Specifies the value to which the **max_area** attribute is to be set. The units of *area* must be consistent with the units in the technology library used during optimization.

DESCRIPTION

This command sets the **max_area** attribute to *area* on the current design. The **max_area** attribute represents the target area of the design and is used by the **compile** command to calculate area cost of the design.

This command is only supported in DC Expert with the **compile** command.

By default, **compile** prioritizes total negative slack (TNS) above area. This means that **compile** does not create new delay violations or worsen existing delay violations on a path that has negative delay slack in order to improve area.

When the **-ignore_tns** option is used, the **ignore_tns** attribute is set on the design. When this attribute is set, **compile** prioritizes area above TNS. This means that **compile** may increase delay violations at an endpoint in order to improve area, as long as the new delay violation is smaller than the delay violation on the endpoint of the most critical path in the same path group.

If you specify **max_area** more than once on a design, the most recent *area* is used and the earlier values are removed.

If the **set_max_area** command is used without the **-ignore_tns** option, then the **ignore_tns** attribute (if any) on the design is deleted.

To undo **set_max_area**, use the **remove_attribute** or **reset_design** command.

Use the **report_area** command to get a detailed summary of area of the design. Use the **report_constraint** command to show area cost of the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the target area for the current design to 0.0 (zero). This causes **compile** to create a design that has been optimized for the smallest possible size.

```
prompt> set_max_area 0.0
```

SEE ALSO

- compile(2)
- current_design(2)
- remove_attribute(2)
- report_area(2)
- report_constraint(2)
- reset_design(2)

set_max_capacitance

Sets the **max_capacitance** attribute to a specified value on the specified clocks, ports and designs.

SYNTAX

```
int set_max_capacitance
    capacitance_value
    [-data_path]
    [-clock_path]
    object_list
```

ARGUMENTS

capacitance_value

Specifies a value to which the **max_capacitance** attribute is to be set. *capacitance_value* must be expressed in units consistent with the technology library used during optimization. For example, if the library specifies capacitance values in picofarads, then *capacitance_value* must also be expressed in picofarads.

-data_path

Restricts the scope of the command to only the pins that are in datapaths launched by the clocks specified in the *object_list*. This option is valid only when the specified object is a clock.

-clock_path

Restricts the scope of the command to only the pins that are in clock paths of the clocks specified in the *object_list*. This option is valid only when the specified object is a clock.

object_list

Specifies a list of names of specified clocks, ports, or designs on which the **max_capacitance** attribute is to be set.

DESCRIPTION

The command sets the **max_capacitance** attribute to a specified value on the specified clocks, ports or designs. Tool attempts to ensure that the capacitance value for a net is less than the specified value. The maximum capacitance value for a net is defined as the least of the maximum capacitance values of the cell pins and design ports on that net.

In cases where both a global maximum capacitance value is set on a design and a local value is set on a port, tool attempts to meet the smaller (more restrictive) value.

If **max_capacitance** attributes are already specified in a technology library (implicit constraints), tool automatically tries to meet them.

By default, a clock, port or design has no **max_capacitance** constraint. You can't set a **max_capacitance** attribute on an output or bidirectional port.

The maximum capacitance constraint is a design rule constraint, which has higher priority over optimization constraints. Thus, the tool gives priority to maximum capacitance constraint, even if it adversely affects optimization constraints on a design. The **max_delay** and **max_area** are optimization constraints, whereas **max_capacitance**, **max_fanout** and **max_transition** are design rule constraints. Design rule constraints reflect those technology-specific restrictions that must be met for a design to function correctly. Optimization constraints reflect goals and restrictions that are desirable, but not crucial for the operation of a design. The tool attempts to meet all constraints placed on a design, but gives priority to design rule constraints during optimization.

When you specify a clock object with the **set_max_capacitance** command, by default, the constraint applies to all pins in the datapaths launched by the clock and to all pins in the clock network. To restrict the scope of the command to either datapaths or clock paths, use the **-data_path** or **-clock_path** option.

To get information about optimization and design rule constraints, use **report_constraint**; to get information on the current port settings, use **report_port**. To remove the **max_capacitance** attribute from a port or a design, use **remove_attribute**.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example sets a maximum capacitance value of 2.0 units on the port named late_riser:

```
prompt> set_max_capacitance 2.0 [get_ports late_riser]
```

The following example specifies a maximum capacitance value of 2.0 units for the TEST design:

```
prompt> set_max_capacitance 2.0 [current_design]
```

The following example specifies a maximum capacitance value of 2.0 for the clock paths of clk1 and the datapaths launched by clk1:

```
prompt> set_max_capacitance 2.0 [get_clocks clk1]
```

The following example specifies a maximum capacitance value of 2.0 for the datapaths launched by clk1:

```
prompt> set_max_capacitance 2.0 -data_path [get_clocks clk1]
```

The following example shows the **-data_path** and **-clock_path** options are invalid when the object is not a clock:

```
prompt> set_max_capacitance 2.0 -data_path [get_port port1]
Error: Cannot specify -data_path/-clock_path without clock object. (CMD-003)
prompt> set_max_capacitance 2.0 -clock_path [current_design]
Error: Cannot specify -data_path/-clock_path without clock object. (CMD-003)
```

SEE ALSO

[all_inputs\(2\)](#)

all_outputs(2)
characterize(2)
remove_attribute(2)
report_constraint(2)
report_port(2)
reset_design(2)
set_min_capacitance(2)

set_max_delay

Specifies a maximum delay target for paths in the current design.

SYNTAX

```
status set_max_delay
    delay_value
    [-rise | -fall]
    [-from from_list
     | -rise_from rise_from_list
     | -fall_from fall_from_list]
    [-through through_list
     | -rise_through rise_through_list
     | -fall_through fall_through_list]
    [-to to_list
     | -rise_to rise_to_list
     | -fall_to fall_to_list]
    [-group_path group_name]
    [-reset_path]
    [-comment comment_string]
    [-ignore_clock_latency]
```

Data Types

<i>delay_value</i>	float
<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list
<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list
<i>group_name</i>	string
<i>comment_string</i>	string

ARGUMENTS

delay_value

Specifies the value of the desired maximum delay for paths between start and end points. You must express *delay_value* in the same units as the technology library used during optimization. If a path startpoint is on a sequential device, clock skew is included in the computed delay. If a path startpoint has an input delay specified, that delay value is added to the path delay. If a

path endpoint is on a sequential device, clock skew and library setup time are included in the computed delay. If the endpoint has an output delay specified, that delay is added into the path delay.

-rise | -fall

Specifies whether endpoint rising or falling delays are constrained. If you don't specify either, both rising and falling delays are constrained.

-from *from_list*

Specifies a list of path startpoints (port, pin, clock, or cell names) of the current design. If you specify a clock, all path startpoints related to that clock are affected. If you specify a cell name, one path startpoint on that cell is affected. All paths from these startpoints to the endpoints in the *to_list* are constrained to *delay_value*. If you don't specify *to_list*, all paths from *from_list* are affected. This list cannot include output ports. If you include more than one object, you must enclose the objects in quotation marks (") or braces ({}).

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through *through_list*

Determines a list of path throughpoints (port, pin, or leaf cell names) of the current design. The max delay value applies only to paths that pass through one of the points in the *through_list*. If you include more than one object, you must enclose the objects in quotation marks (") or braces ({}). If you specify the **-through** option multiple times, the max delay values apply to paths that pass through a member of each *through_list* in the order in which the lists were given. In other words, the path must first pass through a member of the first *through_list*, then through a member of the second list, and so on, for every through list specified. If you use the **-through** option in combination with the **-from** or **-to** option, max delay applies only if the **-from** or **-to** conditions are satisfied and the **-through** conditions are satisfied.

-rise_through *rise_through_list*

Same as the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than once in a single command invocation as with the **-through** option.

-fall_through *fall_through_list*

Same as the **-through** option, but applies only to paths with a falling transition at the specified objects. You can specify **-fall_through** more than once in a single command invocation as with the **-through** option.

-to *to_list*

Specifies a list of path endpoints (port, clock, cell, or pin names) of the current design. All paths to the endpoints in the *to_list* are constrained to *delay_value*. If you don't specify a *from_list*, all paths to *to_list* are affected. This list cannot include input ports. If you include more than one object, you must enclose the objects in quotation marks (") or braces ({}). If you specify a cell, one path endpoint on that cell is affected. If you specify a clock, all path endpoints related to that clock are affected.

-rise_to *rise_to_list*

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-fall_to fall_to_list

Same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-group_path group_name

Specifies the name of the group to which the paths are to be added. If the group does not already exist, it is created. This is equivalent to separately specifying the **group_path** command with the **-name group_name -from from_list -to to_list** options in addition to specifying the **set_max_delay** command. If you do not specify this option, the existing path grouping is not changed.

-reset_path

Removes existing point-to-point exception information on the specified paths. Only information of the same rise/fall setup/hold type is reset. This is equivalent to using the **reset_path** command with similar options before executing the **set_max_delay** command.

-comment comment_string

Allows the command to accept a comment string. The tool honors the annotation and preserves it with the SDC object so that the exact string is written out when the constraint is written out when you use the **write_sdc** or **write_script** command. The comment remains intact through the synthesis, place-and-route, and timing-analysis flows.

DESCRIPTION

Specifies the desired maximum delay for paths in the current design. This command specifies that the maximum path length for any startpoint in *from_list* to any endpoint in *to_list* must be less than *delay_value*.

Individual maximum delay targets are automatically derived from clock waveforms and port input or output delays. For more information, see the **create_clock**, **set_input_delay**, and **set_output_delay** man pages.

The optimization cost of the design depends on how path groups have been specified. For more information, see the **group_path** man page.

The **set_max_delay** command is a point-to-point timing exception command; that is, it overrides the default single-cycle timing relationship for one or more timing paths. Other point-to-point timing exception commands include the **set_multicycle_path**, **set_min_delay**, and **set_false_path** commands.

If a path satisfies multiple timing exceptions, the following rules are applied to determine which exceptions take effect. Rules referring to **-from** apply equally to **-rise_from** and **-fall_from**, and similarly for the rise and fall options of **-through** and **-to**.

1. Two **group_path** commands can conflict with each other. But a **group_path** exception by itself does not conflict with another type of exception. Thus, the remaining rules apply for two **group_path** exceptions or two non-**group_path** exceptions.
2. If both exceptions are **set_false_paths**, no conflict occurs.
3. If one exception is a **set_max_delay** and the other is a **set_min_delay**, no conflict occurs.
4. If one exception is a **set_multicycle_path -hold** and the other is **set_multicycle_path -setup**, no conflict occurs.
5. If one exception is a **set_false_path** and the other is not, the **set_false_path** takes precedence.
6. If one exception is a **set_max_delay** and the other is not, the **set_max_delay** takes precedence.
7. If one exception is a **set_min_delay** and the other is not, the **set_min_delay** takes precedence.

8. If one exception has a -from pin or -from cell and the other does not, the former takes precedence.
9. If one exception has a -to pin or -to cell and the other does not, the former takes precedence.
10. If one exception has any -through points and the other does not, the former takes precedence.
11. If one exception has a -from clock and the other does not, the former takes precedence.
12. If one exception has a -to clock and the other does not, the former takes precedence.
13. The exception with the more restrictive constraint then takes precedence. For **set_max_delay** and **set_multicycle_path -setup**, this is the constraint with the lower value. For **set_min_delay** and **set_multicycle_path -hold**, it is the constraint with the higher value.

The value of a **max_rise_delay** attribute cannot be less than that of a **min_rise_delay** attribute on the same path (and similarly for fall attributes). If this occurs, the old attribute is removed.

Note: Specifying a **min_delay** or **max_delay** to a pin that is not a path endpoint places an implicit **dont_touch** attribute on that cell.

The **-group_path** option modifies the path grouping. Path grouping affects the maximum delay cost function. The worst violator within each group adds to the cost. For optimization, grouping some paths separately can improve their delay cost, but it might also increase design area, and compile time.

Use the **report_timing_requirements** command to list the **max_delay**, **min_delay**, **multicycle_path**, and **false_path** information for the design. Use **report_path_group** to list the path groups which are defined.

To undo **set_max_delay**, use **reset_path**.

To modify paths grouped with the **-group_path** option, use the **group_path** command to place the paths in another group or the default group.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example shows how to optimize the design so that any delay path to a port named Y is less than 10 units.

```
prompt> set_max_delay 10.0 -to {Y}
```

This example shows how to specify that all paths from cell ff1a or ff1b that pass through cell u1 and end at cell ff2e must be less than 15.0 units.

```
prompt> set_max_delay 15.0 -from {ff1a ff1b} -through {u1} -to {ff2e}
```

This example shows how to specify that all paths to endpoints clocked by PHI2 must be less than 8.5 units.

```
prompt> set_max_delay 8.5 -to [get_clocks PHI2]
```

This example shows how to set a requirement that all paths leading to ports named busA[*] have a delay of less than 5.0 and are included in the path group named busA.

```
prompt> set_max_delay 5.0 -to "busA[*]" -group_path "busA"
```

This example shows how to set a maximum delay requirement of 3.0 for all paths that first pass through either pin u1/Z or u2/Z then pass through pin u5/Z or u6/Z.


```
prompt> set_max_delay 3.0 -through {u1/Z u2/Z} -through {u5/Z u6/Z}
```

This example specifies that all timing paths from pin ff1/CP to pin ff2/D that rise through at least one of pins U1/Z and U2/Z and fall through at least one of pins U3/Z and U4/C must have delays less than 8.0 units.

```
prompt> set_max_delay 8.0 -from {ff1/CP} -to {ff2/D} \  
-rise_through {U1/Z U2/Z} -fall_through {U3/Z U4/C}
```

SEE ALSO

- compile(2)
- create_clock(2)
- group_path(2)
- report_constraint(2)
- report_path_group(2)
- reset_design(2)
- reset_path(2)
- set_false_path(2)
- set_input_delay(2)
- set_min_delay(2)
- set_multicycle_path(2)
- set_output_delay(2)

set_max_fanout

Sets the **max_fanout** attribute to a specified value on specified input ports and designs.

SYNTAX

```
status set_max_fanout  
    fanout_value  
    object_list
```

ARGUMENTS

fanout_value

Specifies the value to which the **max_fanout** attribute is to be set; that is, the maximum fanout value.

object_list

Specifies a list of input ports and/or designs on which the **max_fanout** attribute is to be set.

DESCRIPTION

Sets the **max_fanout** attribute on the specified input ports and/or designs. **compile** attempts to ensure that the sum of the **fanout_load** attributes for input pins on nets driven by the specified ports or all nets in the specified design is less than the given value. There is no maximum value placed by default, but if there is a **max_fanout** attribute already specified in a technology library (an implicit constraint), **compile** attempts to meet it.

In cases where both a global fanout limit is set on a design, and a local value is set on a port, **compile** attempts to meet the smaller (more restrictive) value.

In general, design rule constraints reflect technology-specific constraints that *must* be met for a design to function correctly. Optimization constraints are design goals and restrictions that are desirable but not crucial for the operation of a design. **compile** attempts to meet all constraints placed on a design, but it gives priority to design rule constraints in the optimization process.

The **max_fanout** attribute specifies Maximum Fanout, a *design rule constraint*. Thus, **compile** gives precedence to fixing a **max_fanout** violation, even if it adversely affects *optimization constraints* (for example, Maximum Delay and Maximum Area).

Use **report_constraint** to get information about optimization and design rule constraints.

To remove a maximum fanout from a port or design, use **remove_attribute**. **reset_design** removes all attributes, including **max_fanout**, from the current design.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example sets a maximum fanout of 20.0 units on the port named going_places:

```
prompt> set_max_fanout 20.0 going_places
```

In the following example, the **max_fanout** attribute is set to 18.5 on the TEST design:

```
prompt> set_max_fanout 18.5 TEST
```

SEE ALSO

- all_inputs(2)
- remove_attribute(2)
- report_constraint(2)
- report_port(2)
- reset_design(2)
- set_fanout_load(2)

set_max_time_borrow

Sets the **max_time_borrow** attribute to a specified value on clocks, latch cells, data pins, or clock (enable) pins, to constrain the amount of time borrowing possible for level-sensitive latches.

SYNTAX

```
status set_max_time_borrow  
    delay_value  
    object_list
```

ARGUMENTS

delay_value

Specifies the value to which the **max_time_borrow** attribute is to be set; defines the desired limit of time borrowing on the latches specified in **object_list**. *delay_value* must be between zero and the default maximum derived from the waveform. By default, the maximum is derived from the ideal clock waveform driving each latch, and is equal to (closing_edge - open_edge). Library setup and data-to-Q propagation times are automatically taken into account. *delay_value* is assumed to be in the same units as those in the technology library used during optimization.

object_list

Specifies a list of objects in the current design for which time borrowing is to be limited to *delay_value*. The objects can be clocks, latch cells, data pins, or clock (enable) pins. If a cell is specified, all enable pins on that cell are affected. Note that if the **max_time_borrow** attribute is set on a cell and the cell is replaced during optimization, the attribute is moved from the cell to the enable pin.

DESCRIPTION

Sets the **max_time_borrow** attribute to **delay_value** to constrain the amount of time borrowing possible for level-sensitive latches. To meet delay targets, **set_max_time_borrow** prevents automatic use of all or part of the enabling clock pulse on a latch.

To get **max_time_borrow** attributes on the design, use **get_attribute**.

To undo **set_max_time_borrow**, use **remove_attribute**.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example restricts time borrowing on latches latch1a and latch2f to 4.0 units.

```
prompt> set_max_time_borrow 4.0 {latch1a latch2f}
```

The following example specifies that no time borrowing will take place on the latch latch1c.

```
prompt> set_max_time_borrow 0.0 {latch1c}
```

SEE ALSO

`remove_attribute(2)`

set_max_transition

Sets the maximum transition time for the specified clocks, ports, or designs.

SYNTAX

```
status set_max_transition  
    transition_value  
    [-data_path]  
    [-clock_path]  
    object_list
```

Data Types

transition_value float
object_list list or collection

ARGUMENTS

transition_value

Specifies the maximum transition time for the specified clocks, ports, or designs. It sets the **max_transition** attribute to the specified *transition_value* on the specified objects. The value must be expressed in units consistent with the those in the technology library used during optimization. For example, if the library specifies drive values in kilohms and load values in picofarads, then *transition_value* must be expressed in nanoseconds.

-data_path

Restricts the scope of the command to only the pins that are in data paths launched by the clocks specified in the *object_list*.

-clock_path

Restricts the scope of the command to only the pins that are in clock paths of the clocks specified in the *object_list*.

object_list

Specifies the names of the clocks, ports, or designs affected by the command. The tool seeks to reduce transition times on the nets associated with these objects to no more than the time value specified by *transition_value*.

DESCRIPTION

This command sets the maximum transition time, a design rule constraint, on specified clocks, ports, or designs. The tool attempts to ensure that the transition time for a net is less than the specified value. The maximum transition time for a net is defined as the least

of the maximum transition times of the cell pins and design ports on that net.

In cases where a global maximum transition time is set on a design and a local value is set on a port or clock group, the tool attempts to meet the more restrictive (smaller) value.

If the **max_transition** attributes are already specified in a technology library (implicit constraints), the tool attempts to meet them.

By default, no restriction is placed on the transition time for an input or output port, so the maximum transition time for the driven net is determined by the maximum transition time (if any) of the driven cell inputs or the driving cell output. Assigning a smaller maximum transition time to a port reduces the maximum transition time for the net; thus, the design constraints are more restrictive.

The maximum transition constraint is a design rule constraint, which has higher priority over optimization constraints. Thus, the tool gives priority to maximum transition constraint, even if it adversely affects optimization constraints on a design. Both **max_delay** and **max_area** are optimization constraints, whereas **max_fanout** and **max_transition** are design rule constraints. Design rule constraints reflect those technology-specific restrictions that must be met for a design to function correctly. Optimization constraints reflect goals and restrictions that are desirable, but not crucial for the operation of a design. The tool attempts to meet all constraints placed on a design, but gives priority to design rule constraints during optimization.

When you specify a clock object with the **set_max_transition** command, by default, the constraint applies to all pins in the data paths launched by the clock and to all pins in the clock network. To restrict the scope of the command to either data paths or clock paths, use the **-data_path** or **-clock_path** option.

Use the **report_constraint** command to get information about optimization and design rule constraints.

Use the **remove_attribute** command to remove a maximum transition time constraint from a port, design, or clock object. When you use the **remove_attribute** command to remove the maximum transition constraint from a clock object, the command removes the constraint from both the data paths and clock paths related to the specified clock.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example sets a maximum transition time of 2.0 units on the port named `late_riser`:

```
prompt> set_max_transition 2.0 late_riser
```

The following example specifies a maximum transition time of 2.0 units for the design `TEST`:

```
prompt> set_max_transition 2.0 TEST
```

The following example specifies a maximum transition time of 2.0 units for the clock paths of `clk1` and the data paths launched by `clk1`:

```
prompt> set_max_transition 2.0 [get_clocks clk1]
```

The following example specifies a maximum transition time of 2.0 units for the data paths launched by `clk1`:

```
prompt> set_max_transition 2.0 -data_path [get_clocks clk1]
```

SEE ALSO

all_inputs(2)
all_outputs(2)
characterize(2)
current_design(2)
remove_attribute(2)
report_constraint(2)
reset_design(2)

set_message_info

Set some information about diagnostic messages.

SYNTAX

```
string set_message_info  
-id message_id [-limit max_limit] -stop_on  
[-stop_off]
```

Data Types

```
message_id  string  
max_limit   integer
```

ARGUMENTS

-id message_id

Information is to be set for the given *message_id*. The message must exist. Although different constraints allow different message types, no constraint allows severe or fatal messages.

-limit max_limit

Set the maximum number of occurrences for *message_id*. This is an integer greater than or equal to zero. If you set it to zero, that means the number of occurrences of the message is unlimited. Messages which occur after a limit is reached are automatically suppressed.

-stop_on

Force Tcl error if message is emitted.

-stop_off

Turn off a previous -stop_on directive

DESCRIPTION

The **set_message_info** command sets constraints on diagnostic messages (typically error, warning, and informational messages).

Currently, you can set a upper limit for the number of occurrences of a message. You can set this to zero to indicate that there is no limit. You can retrieve the current limit for a message using the **get_message_info** command. *When the limit is exceeded, all future occurrences of the message are automatically suppressed. A count of total occurrences (including those suppressed)*

can be retrieved using `get_message_info`.

EXAMPLES

The following example uses **set_message_info** to set a limit on the number of APP-027 messages to 100. When the 101st APP-027 message is about to be issued, you will be warned that the limit has been exceeded, and that all future occurrences will be suppressed.

```
prompt> set_message_info -id APP-027 -limit 100
prompt> do_command
Warning: can't find node U27.1 (APP-027)
Warning: can't find node U27.2 (APP-027)
Warning: can't find node U27.3 (APP-027)
...
Warning: can't find node U27.100 (APP-027)
Note - message 'APP-027' limit (100) exceeded. Remainder will be suppressed.
1
```

SEE ALSO

`get_message_info(2)`
`get_message_ids(2)`
`print_message_info(2)`
`suppress_message(2)`

set_message_severity

Sets the severity level of violation messages for boundary-scan compliance checking.

SYNTAX

```
int set_message_severity  
  -names message_tags  
  severity
```

Data Types

```
message_tags  list  
severity      string
```

ARGUMENTS

-names *message_tags*

Specifies the list of messages **message_tags** whose severity needs to be changed.

severity

Specifies the severity to use for the messages *message_tags*. Valid values are **error**, **warning**, and **ignore**.

DESCRIPTION

The **set_message_severity** command identifies the set of boundary-scan compliance violation messages whose severity needs to be changed. This can allow exceptions to the compliance checking performed by the **check_bsd** command. Both the **-names** and *severity* arguments are required.

The **set_message_severity** command can also be used to reset the severity of a message back to its original type.

EXAMPLES

The following example specifies the severity of the TEST-813 and TEST-893 messages to be warnings:

```
prompt> set_message_severity -names {TEST-813 TEST-893} warning
```

The following example resets the severity of the TEST-893 message to be an error:

```
prompt> set_message_severity -names {TEST-893} error
```

set_min_capacitance

Sets the **min_capacitance** attribute to a specified value on specified input ports in the current design.

SYNTAX

```
status set_min_capacitance  
      capacitance_value  
      object_list
```

Data Types

```
capacitance_value  float  
object_list       list
```

ARGUMENTS

capacitance_value

Specifies a value to which the **min_capacitance** attribute is to be set. *capacitance_value* must be expressed in units consistent with the technology library used during optimization. For example, if the library specifies capacitance values in picofarads, then *capacitance_value* must also be expressed in picofarads.

object_list

Specifies a list of names of the input and/or bidirectional ports on which the **min_capacitance** attribute is to be set.

DESCRIPTION

Sets the **min_capacitance** attribute to *capacitance_value* on the specified input ports. **compile** attempts to ensure that the load driven by the input port is not reduced below *capacitance_value*. The minimum capacitance value for a net is defined as the greatest of the minimum capacitance values of the driving cell pins and ports on a net.

If **min_capacitance** attributes are already specified in a technology library (implicit constraints), **compile** automatically tries to meet them.

By default, a port has no **min_capacitance** constraint.

min_capacitance, along with **max_fanout** and **max_transition**, is a *design rule constraint*; **max_delay** and **max_area** are optimization constraints. Design rule constraints reflect those technology-specific restrictions that *must* be met for a design to function correctly, while optimization constraints reflect goals and restrictions that are desirable, but not crucial for the operation of a design. Design Compiler attempts to meet all constraints placed on a design, but gives priority to design rule constraints in the optimization process. Therefore, **compile** gives preference to **min_capacitance** and other design rule constraints, even if they

adversely affect optimization constraints on a design.

To get information about optimization and design rule constraints, use **report_constraint**; to get information on the current port settings, use **report_port**. To remove the **min_capacitance** attribute from a port, use **remove_attribute**.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example sets a minimum capacitance value of 12.0 units on the port named high_drive:

```
prompt> set_min_capacitance 12.0 high_drive
```

SEE ALSO

- all_inputs(2)
- characterize(2)
- compile(2)
- remove_attribute(2)
- report_constraint(2)
- report_port(2)
- reset_design(2)
- set_max_capacitance(2)

set_min_delay

Specifies a minimum delay target for paths in the current design.

SYNTAX

```
status set_min_delay
    delay_value
    [-rise | -fall]
    [-from from_list
     | -rise_from rise_from_list
     | -fall_from fall_from_list]
    [-through through_list
     | -rise_through rise_through_list
     | -fall_through fall_through_list]
    [-to to_list
     | -rise_to rise_to_list
     | -fall_to fall_to_list]
    [-reset_path]
    [-comment comment_string]
    [-ignore_clock_latency]
```

Data Types

<i>delay_value</i>	float
<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list
<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list
<i>comment_string</i>	string

ARGUMENTS

delay_value

Specifies the value of the desired minimum delay for paths between startpoints and endpoints. Express *delay_value* in the same units as the technology library used during optimization. If a path startpoint is on a sequential device, clock skew is included in the computed delay. If a path startpoint has an input external delay specified, that delay value is added to the path delay. If a path endpoint is on a sequential device, clock skew and library setup time are included in the computed delay. If the endpoint has an output external delay specified, that delay is added into the path delay.

-rise | -fall

Specifies whether endpoint rising or falling delays are constrained. If neither is specified, both rising and falling delays are constrained.

-from *from_list*

Lists the path startpoints (port, pin, clock, or cell names) of the current design. If a clock is specified, all path startpoints related to that clock are affected. If you specify a cell name, one path startpoint on that cell is affected. All paths from these startpoints to the endpoints in the *to_list* are constrained to *delay_value*. If you do not specify *to_list*, all paths from *from_list* are affected. This list cannot include output ports. If more than one object is included, enclose the objects in double quotation marks (") or in braces ({}).

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through *through_list*

Lists the path throughpoints (port, pin, or leaf cell names) of the current design. The minimum delay value applies only to paths that pass through one of the points in the *through_list*. If you include more than one object, enclose the objects in double quotation marks (") or in braces ({}). If you specify the **-through** option multiple times, the minimum delay values apply to paths that pass through a member of each *through_list* in the order the lists were given. The path must first pass through a member of the first *through_list*, then through a member of the second list, and so on for every through list specified. If you use the **-through** option in combination with the **-from** or **-to** options, the minimum delay applies only if the **-from** or **-to** conditions are satisfied and the **-through** conditions are satisfied.

-rise_through *rise_through_list*

Same as the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than once in a single command invocation as with the **-through** option.

-fall_through *fall_through_list*

Same as the **-through** option, but applies only to paths with a falling transition at the specified objects. You can specify **-fall_through** more than once in a single command invocation as with the **-through** option.

-to *to_list*

Lists the path endpoints (port, clock, cell, or pin names) of the current design. All paths to the endpoints in the *to_list* are constrained to *delay_value*. If you do not specify *from_list*, all paths to *to_list* are affected. This list cannot include input ports. If more than one object is included, enclose the objects in either double quotation marks (") or in braces ({}). If you specify a cell, one path endpoint on that cell is affected. If you specify a clock, all path endpoints related to that clock are affected.

-rise_to *rise_to_list*

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-fall_to *fall_to_list*

Same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints

clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-reset_path

Removes existing point-to-point exception information on the specified paths. Only information of the same rise and fall setup or hold type is reset. This is equivalent to using the **reset_path** command with similar arguments before the **set_min_delay** command is issued.

-comment *comment_string*

Allows the command to accept a comment string. The tool honors the annotation and preserves it with the SDC object so that the exact string is written out when the constraint is written out when you use the **write_sdc** or **write_script** command. The comment remains intact through the synthesis, place-and-route, and timing-analysis flows.

DESCRIPTION

This command sets the minimum delay target for paths in the current design. Minimum delay is considered as an optimization constraint by the **compile** command. If a path violates the requirement given in a **set_min_delay** command, **compile** adds delay to fix the violation if maximum delay cost is not increased. You can prioritize minimum delay cost using the **set_cost_priority** command.

The value of a **min_rise_delay** attribute can never be greater than that of a **max_rise_delay** attribute for the same path (and similarly for fall attributes). If this occurs, the old attribute is removed.

Individual minimum delay targets are automatically derived from clock waveforms and port input or output delays. For more information, refer to the **create_clock**, **set_input_delay**, and **set_output_delay** command man pages.

The **set_min_delay** command is a point-to-point timing exception command. It overrides the default single-cycle timing relationship for one or more timing paths. Other point-to-point timing exception commands include **set_multicycle_path**, **set_max_delay**, and **set_false_path**.

If a path satisfies multiple timing exceptions, the following rules determine which exceptions take effect. Rules referring to **-from** apply equally to **-rise_from** and **-fall_from**, and similarly for the rise and fall options of **-through** and **-to**.

1. Two **group_path** commands can conflict with each other, but a **group_path** exception alone does not conflict with another type of exception. Therefore, the remaining rules apply for two **group_path** exceptions, or two non **group_path** exceptions.
2. If both exceptions are **set_false_path**, there is no conflict.
3. If one exception is a **set_max_delay** and the other is a **set_min_delay**, there is no conflict.
4. If one exception is a **set_multicycle_path -hold** and the other is a **set_multicycle_path -setup**, there is no conflict.
5. If one exception is a **set_false_path** and the other is not, the **set_false_path** takes precedence.
6. If one exception is a **set_max_delay** and the other is not, the **set_max_delay** takes precedence.
7. If one exception is a **set_min_delay** and the other is not, the **set_min_delay** takes precedence.
8. If one exception has a **-from pin** or **-from cell** and the other does not, the one with the **-from pin** takes precedence.
9. If one exception has a **-to pin** or **-to cell** and the other does not, the one with the **-to pin** takes precedence.
10. If one exception has any **-through** points and the other does not, the one with **-through** points takes precedence.
11. If one exception has a **-from clock** and the other does not, the one with the **from clock** takes precedence.

12. If one exception has a **-to clock** and the other does not, the one with the **-to clock** takes precedence.
13. The exception with the more restrictive constraint then takes precedence. For **set_max_delay** and **set_multicycle_path -setup**, this is the constraint with the lower value. For **set_min_delay** and **set_multicycle_path -hold**, it is the constraint with the higher value.

To remove information set by **set_min_delay**, use the **reset_path** or **reset_design** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

In the following example, the **set_min_delay** command requires that any delay path that passes through the U1 cell and ends at the Y port is greater than 12.5 time units:

```
prompt> set_min_delay 12.5 -through U1 -to Y
```

The following example specifies that all paths from A1 and A2 to Z5 must be greater than 4.0 units:

```
prompt> set_min_delay 4.0 -from {A1 A2} -to Z5
```

The following example command sets a minimum delay requirement of 3.0 for all paths that first pass through either pin u1/Z or u2/Z and then pass through pin u5/Z or u6/Z:

```
prompt> set_min_delay 3.0 -through {u1/Z u2/Z} -through {u5/Z u6/Z}
```

The following example specifies that all timing paths from pin ff1/CP to pin ff2/D that rise through at least one of pins U1/Z and U2/Z and fall through at least one of pins U3/Z and U4/C must have delays greater than 3.0 units.

```
prompt> set_min_delay 3.0 -from ff1/CP -to ff2/D \  
-rise_through {U1/Z U2/Z} -fall_through {U3/Z U4/C}
```

SEE ALSO

compile(2)
report_constraint(2)
reset_design(2)
reset_path(2)
set_cost_priority(2)
set_fix_hold(2)
set_max_delay(2)

set_min_library

Sets an alternate library to use for minimum delay analysis.

SYNTAX

```
int set_min_library  
    max_library  
    -min_version min_library | -none
```

Data Types

<i>max_library</i>	string
<i>min_library</i>	string

ARGUMENTS

max_library

Specifies the library file name to use as a `link_library` or `target_library`. This library is used for maximum delay analysis only. Do not include the path to the library file name.

`-min_version` *min_library*

Specifies the library file name that corresponds to the *max_library* and uses it for minimum delay analysis. The *min_library* should not be specified in the `link_library` or `target_library`. Do not include the path to the library file name.

`-none`

Unsets the setting of a minimum library.

DESCRIPTION

The **set_min_library** command creates a minimum/maximum relationship between two library files. The *max_library* is used for maximum delay analysis and the *min_library* is used for minimum delay analysis.

Whenever the tool computes a minimum delay value, it first consults the library cell from the max library. If a library cell exists with the same name, the same pins, and the same timing arcs in the min library, the timing information from the min library is used. If the tool cannot find a matching cell in the min library, the max library cell is used.

Do not include the path to the *max_library* nor *min_library* file name. Specify the path to those library files in `$search_path`.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example is a typical usage of **set_min_library**:

```
prompt> set link_library "LIB_WC_COM.db"
prompt> set target_library $link_library
prompt> set_min_library LIB_WC_COM.db -min_version LIB_BC_COM.db
prompt> set_operating_conditions -max WC_COM -max_library LIB_WC_COM \
    -min BC_COM -min_library LIB_BC_COM
prompt> report_timing -delay max
prompt> report_timing -delay min
```

SEE ALSO

link(2)
set_operating_conditions(2)
link_library(3)
target_library(3)

set_min_pulse_width

Sets a minimum pulse width constraint for clocks or clock pins.

SYNTAX

```
status set_min_pulse_width
  [-low]
  [-high]
  value
  object_list
```

Data Types

value float
object_list list

ARGUMENTS

-low

Indicates that the minimum pulse width constraint specified by *value* applies only to low clock signal levels. If you do not specify the *-low* or *-high* option, both low and high pulses of clock signals are constrained.

-high

Indicates that the minimum pulse width constraint specified by *value* applies only to high clock signal levels. If you do not specify the *-low* or *-high* option, both low and high pulses of clock signals are constrained.

value

A positive floating-point value that specifies the minimum pulse width check to be applied to clock signals.

object_list

Specifies a list of clocks or clock pins in the current design to which the minimum pulse width check is applied. If you specify a clock, the constraint is applied to all clock pins connected to the clock.

DESCRIPTION

The **set_min_pulse_width** command specifies a minimum pulse width check to be applied to clock signals at clock pins of sequential devices. This value overrides the default values for clock tree minimum pulse width checks. Use this command for sequential clock pin pulse width checks to add a more restrictive value than the one specified in library. This setting is applied in all

scenarios.

To generate a report of pulse width constraints, use **report_constraint** or **report_min_pulse_width**. To remove minimum pulse width constraints set by **set_min_pulse_width**, use the **remove_min_pulse_width** command.

EXAMPLES

The following example sets a minimum pulse width requirement of 2.0 for both low and high pulses of clock signals.

```
prompt> set_min_pulse_width 2.0 [get_clocks CK1]
```

The following example sets a minimum pulse width requirement of 2.5 for the low pulse of the clock signal on the clock pin reg1/CK.

```
prompt> set_min_pulse_width -low 2.5 reg1/CK
```

SEE ALSO

[remove_min_pulse_width\(2\)](#)
[report_constraint\(2\)](#)
[report_min_pulse_width\(2\)](#)

set_minimize_tree_delay

Sets the **minimize_tree_delay** attribute on a design or designs.

SYNTAX

```
integer set_minimize_tree_delay  
  [true | false]  
  [-design design_list]  
  [design_list]
```

Data Types

design_list list

ARGUMENTS

true | false

Determines the value to which the **minimize_tree_delay** attribute is to be set. The default is **true**.

-design *design_list*

Specifies a list of designs on which to set the **minimize_tree_delay** attribute. The default is the current design.

DESCRIPTION

This command sets the **minimize_tree_delay** attribute on a design or designs, thus determining whether an arithmetic expression tree (for example, an adder chain) is to be restructured to minimize delay during **compile**. If **minimize_tree_delay** is not set, then the value of the **hlo_minimize_tree_delay** environment variable is used. The default value of this variable is **true**. Thus, by default, all expression trees are candidates for tree height minimization if timing constraints are specified. Setting the **minimize_tree_delay** attribute overrides the value of **hlo_minimize_tree_delay** for the design or designs on which the attribute is set.

The restructuring of arithmetic expression trees is based on the arrival times of inputs to the tree. In the case of equal arrival times, a balanced tree is constructed.

Note that even if **minimize_tree_delay** is set to **true**, tree height minimization is not performed if the design has no timing constraints, or if the **hlo_resource_allocation** variable is set to *area_only*, *area_no_tree_balancing*, or *none*.

For more details on tree height minimization, see the *VHDL Compiler Reference Manual* or the *HDL Compiler for Verilog Reference Manual*.

To remove the **minimize_tree_delay** attribute, use the **remove_attribute** command. The **reset_design** command removes all

attributes from a design, including **minimize_tree_delay**.

EXAMPLES

In the following example, **minimize_tree_delay** is enabled on the design named TEST:

```
prompt> set_minimize_tree_delay -design TEST
```

In the following example, **minimize_tree_delay** is enabled on the current design and disabled on the design named OLD:

```
prompt> read TEST
```

```
prompt> set_minimize_tree_delay
```

```
prompt> set_minimize_tree_delay false -design OLD
```

SEE ALSO

- `compile(2)`
- `current_design(2)`
- `remove_attribute(2)`
- `reset_design(2)`
- `design_attributes(3)`
- `hlo_resource_allocation(3)`

set_mode

Selects the mode of a component.

SYNTAX

```
status set_mode  
  [mode_list]  
  [instance_list]
```

Data Types

```
mode_list    list  
instance_list list
```

ARGUMENTS

mode_list

Specifies the active modes for timing analysis. All other modes of the given mode group are implicitly inactive. The specified modes are active on the given instance list.

instance_list

Specifies the cells for which the specified modes are active.

DESCRIPTION

The **set_mode** command is used to select the active mode of a cell that has several functional modes. The other modes in the mode group that are not selected are disabled.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following command selects the READ mode for the RAM instance Uram1.

```
prompt> set_mode READ Uram1
```

SEE ALSO

report_mode(2)

reset_mode(2)

set_model_drive

Sets the **model_drive** attribute to a specified value on specified input or inout ports to set their drive values during synthetic library modeling.

SYNTAX

```
int set_model_drive  
    drive_value  
    port_list
```

Data Types

```
drive_value  float  
port_list    list
```

ARGUMENTS

drive_value

Specifies the value to which the **model_drive** attribute is to be set, thus specifying the estimated drive value for the *port_list* in terms of standard drives of the current technology library. The value must be greater than or equal to zero.

port_list

Specifies a list of input or inout port names on which the **model_drive** attribute is to be set. If more than one port is specified, they must be enclosed in quotes (") or in braces ({}).

DESCRIPTION

Sets the **model_drive** attribute to a specified value on specified input or inout ports when performing synthetic library modeling. Setting the attribute sets the estimated drive of ports in terms of standard drives of the current technology library. A standard drive is equal to the output drive of a typical gate in the current target library.

To view **model_drive** values on ports, use **report_port**.

To remove **model_drive** attributes from ports, use **remove_attribute**. **reset_design** removes all attributes, including **model_drive**.

CAVEATS

This command is valid only during synthetic library modeling, and has **no effect** when run directly from dc_shell. Therefore the command also has no effect when directly elaborating DesignWare parts that contain these directives in their embedded dc_shell scripts. To set the drive in dc_shell use the set_drive command.

EXAMPLES

The following example sets the **model_drive** attribute to 2.0 on ports "A", "B" and "C".

```
prompt> set_model_drive 2.0 {A B C}
```

SEE ALSO

- remove_attribute(2)
- report_port(2)
- reset_design(2)
- set_drive(2)
- set_model_load(2)

set_model_load

Sets the **model_load** attribute to a specified value on specified ports to set their load values during synthetic library modeling.

SYNTAX

```
int set_model_load  
    load_value port_list
```

Data Types

```
load_value    float  
port_list     list
```

ARGUMENTS

load_value

Specifies the value to which the **model_load** attribute is to be set, thus specifying the estimated load value for the *port_list* in terms of standard loads of the current technology library. Must be greater than or equal to zero.

port_list

Specifies a list of output or inout port names on which the **model_load** attribute is to be set. If more than one port is specified, they must be enclosed in quotes (") or in braces ({}).

DESCRIPTION

Sets the **model_load** attribute to a specified value on specified output or inout ports during synthetic library modeling. Setting the attribute sets the estimated load of ports in terms of standard loads of the current technology library. A standard load is equal to the input pin capacitance of a typical gate in the current target library. Load values are stored on ports as the **model_load** attribute.

To view **model_load** values on ports, use **report_port**.

To remove **model_load** attributes from ports, use **remove_attribute**. **reset_design** removes all attributes, including **model_load**.

CAVEATS

This command is valid only during synthetic library modeling, and has **no effect** when run directly from dc_shell. Therefore the command also has no effect when directly elaborating DesignWare parts that contain these directives in their embedded dc_shell scripts. To set the load in dc_shell use the set_load command.

EXAMPLES

The following example sets the **model_load** attribute to 2.0 on ports "A", "B" and "C".

```
prompt> set_model_load 2.0 {A, B, C}
```

SEE ALSO

- find(2)
- remove_attribute(2)
- report_port(2)
- reset_design(2)
- set_load(2)
- set_model_drive(2)

set_model_map_effort

Sets the **model_map_effort** attribute to a specified value on the current design, to specify the relative amount of CPU time to use during synthetic library modeling.

SYNTAX

```
int set_model_map_effort  
    low | medium | high
```

ARGUMENTS

low | medium | high

Specifies the value to which the **model_map_effort** attribute is to be set, which specifies the relative amount of CPU time spent during the mapping phase of modeling. Values of 1, 2, or 3 can also be used.

DESCRIPTION

Sets the **model_map_effort** attribute to a specified value on the current design during synthetic library modeling. A value of *low*, *medium*, or *high* indicates the relative amount of CPU time to be used during the mapping phase of modeling. This command would typically be used in an embedded dc_shell script in a synthetic library part. The value of **model_map_effort** attribute (*low*, *medium*, or *high*) indicates the relative amount of CPU time to be used during the mapping phase of modeling, and overrides the default value stored in the environment variable **synlib_model_map_effort**. If the **model_map_effort** attribute is not set, the value of **synlib_model_map_effort** is used.

CAVEATS

This command is valid only during synthetic library modeling, and has **no effect** when run directly from dc_shell. Therefore the command also has no effect when directly elaborating DesignWare parts that contain these directives in their embedded dc_shell scripts. To set the model_map_effort in dc_shell use the -map_effort option of the compile command.

SEE ALSO

compile(2)
set_model_load(2)
set_model_drive(2)

set_multi_vth_constraint

Limits the usage of low-threshold-voltage cells to a specified percentage of all cells.

SYNTAX

```
status set_multi_vth_constraint
  [-lvth_groups groups]
  [-lvth_percentage percent_value]
  [-cost cell_count | area]
  [-type hard | soft]
  [-include_blackboxes]
  [-reset]
```

Data Types

groups string list
percent_value float

ARGUMENTS

-lvth_groups *groups*

Specifies the threshold-voltage groups considered to be low-threshold-voltage groups. The *groups* argument is a list of threshold-voltage group identifiers, which are names defined by attributes in the Liberty (.lib) descriptions of the cells and cell libraries.

-lvth_percentage *percent_value*

Specifies the maximum allowable percentage of low-threshold-voltage cells for the current design. The *percent_value* argument is a floating-point number between 0 and 100.

-cost cell_count | area

This option is not supported in Design Compiler.

-type hard | soft

Specifies the constraint type, which can be either **hard** or **soft**. The default is **soft**; the tool meets this constraint only if it does not adversely affect the timing constraint.

When this option is set to **hard**, the tool monitors the percentage of low-threshold-voltage cells and reduces their usage to meet this constraint, even at the expense of timing.

-include_blackboxes

Includes black box cells in the percentage calculation. By default, black box cells are excluded from the calculation.

-reset

Removes the low-threshold-voltage percentage constraint from the current design.

DESCRIPTION

The **set_multi_vth_constraint** command identifies the low-threshold-voltage library cell groups and sets an upper limit on the percentage of cells that are allowed to come from these groups. Low-threshold-voltage cells are faster than regular cells but have more leakage and therefore increase power. Limiting the usage of these cells to a specified percentage helps to minimize power.

The **-lvth_groups** option identifies the low-threshold-voltage library groups and the **-lvth_percentage** option specifies the percentage usage constraint.

By default, black box cells are excluded from the percentage calculation. To include black box cells in the calculation, use the **-include_blackboxes** option.

By default, the constraint is a soft constraint, which is not enforced if it adversely affects timing. To make the constraint hard, use the **-type hard** option. In that case, the tool attempts to enforce the constraint, even if timing is adversely affected.

The threshold-voltage group of a cell is defined by the threshold-voltage group of its library cell reference, which is defined by the value of the **threshold_voltage_group** library cell attribute or the **default_threshold_voltage_group** library attribute in the Liberty (.lib) description of the cell or library.

For example, consider a design that contains cells that belong to the following threshold-voltage groups: lvt, xlvt, svt, and hvt. If lvt and xlvt are specified as the low-threshold-voltage groups, the percentage of the low-threshold-voltage cells in the design is:

$$100 * (\text{lvt and xlvt cells used}) / (\text{total cells used})$$

To report the number and percentage of cells used in the design from each threshold voltage group, use the **report_threshold_voltage_group** command.

When you use the **set_multi_vth_constraint** command without any options, it reports the multiple-threshold-voltage constraint previously set on the design.

To remove the current multiple-threshold-voltage constraint from the design, use the **-reset** option.

The **-type soft** option is supported only in topographical mode.

The **set_multi_vth_constraint** command is incompatible with the following commands:

- set_max_leakage_power
- set_max_dynamic_power
- set_leakage_optimization
- set_dynamic_optimization

If any of these commands are used with the **set_multi_vth_constraint** command, they are ignored.

Multicorner-Multimode Support

At least one scenario must be a leakage scenario for the **set_multi_vth_constraint** command to work.

EXAMPLES

The following example sets a multiple-threshold-voltage soft constraint of a maximum of 20 percent low-threshold-voltage cells. The calculation is based on cell count and includes black box cells. Cells that belong to the xlvt and lvt threshold-voltage groups are considered low-threshold-voltage cells.

For example, if 10 percent of the cells belong to the xlvt group and 5 percent belong to the lvt group, this constraint is met. This is because only 15 percent of the cells belong to low-threshold-voltage groups. However, if 15 percent of the cells belong to the xlvt group and 10 percent belong to the lvt group, the constraint is not met because 25 percent of the cells in the design belong to low-threshold-voltage groups.

```
prompt> set_multi_vth_constraint -lvth_groups {xlvt lvt} \  
-lvth_percentage 20.0 -include_blackboxes
```

The following example removes the multiple-threshold-voltage constraint set on the current design.

```
prompt> set_multi_vth_constraint -reset
```

SEE ALSO

- `compile_ultra(2)`
- `report_power(2)`
- `report_threshold_voltage_group(2)`

set_multibit_options

Sets the **multibit_mode** and **minimum_multibit_width** attributes to specified values on the current design.

SYNTAX

```
int set_multibit_options  
  [-default]  
  [-mode multibit_mode] [-minimum_width width]  
  [-critical_range range]  
  [-path_groups list_of_pathgoups]
```

Data Types

<i>multibit_mode</i>	string
<i>width</i>	int
<i>range</i>	string

ARGUMENTS

-default

Sets the **multibit_mode** and **minimum_multibit_width** attributes so that **compile** uses their default values.

-mode *multibit_mode*

Specifies the value for the **multibit_mode** attribute.

-minimum_width *width*

Specifies the value for the **minimum_multibit_width** attribute.

-critical_range *range*

Specifies the critical range to be used in order to determine which multibit registers need to be reworked for timing optimization. The range is expected to be a non-negative number or the word default. When the word default is given, the tool will determine an appropriate critical range for each pathgroup based on the state of the timing constraints at the moment of the optimization. When the range is zero, only registers on the critical paths will be considered.

-path_groups *list_of_pathgoups*

Restricts the given impact of the given -critical_range to the path groups specified in the list. This option must go together with the -critical_range, otherwise it is meaningless.

DESCRIPTION

The **-mode** option specifies the value for the **multibit_mode** attribute. The **multibit_mode** attribute value must be one of the following: **non_timing_driven**, **timing_driven**, or **timing_only** or **none**. The default for the **multibit_mode** attribute is **non_timing_driven**.

In the **non_timing_driven** mode, the tool tries to use multibit registers whenever possible. It provides the best possible multibit register inference ratio.

In the **timing_driven** mode, multibit components are mapped to multibit registers only when it does not hurt timing or area QoR.

In the **timing_only** mode, multibit registers are used only when it does not hurt timing QoR even when it hurts area QoR.

If the mode specified is **none**, then no multibit optimizations will take place.

The **-minimum_width** option specifies the value for the **minimum_multibit_width** attribute when an old **multibit_mode** attribute is set. This option has no effect when the **multibit_mode** attribute is set to **non_timing_driven**, **timing_driven**, or **timing_only**.

The **minimum_multibit_width** and the **multibit_mode** attribute values can be checked using the **report_compile_options** command.

The **-critical_range** and **-path_groups** options will trigger the **mbm_timing_opto** optimization step during incremental compile. This optimization is only available in SPG mode, so specifying its controlling options in a non-SPG flow will have no impact on the final results. This optimization will attempt to unpack the multibits cells within the critical range of their path groups in order to extract the critical bits as single-bit cells. Then, if the design is not scan stitched or there are external scan multibit cells in the library, the tool will attempt to re-pack the non-critical bits to minimize packing ratio degradation.

EXAMPLES

The following example sets the **multibit_mode** attribute.

```
prompt> set_multibit_options -mode timing_driven
```

The following example shows the attributes after the previous command:

```
prompt> report_compile_options
```

```
*****
Report : compile_options
Design : example
Version: J-2014.09-SP5-5
Date   : Tue Nov 3 22:40:04 2015
*****
```

Design	Compile Option	Value
example	flatten	false
	structure	true
	structure_boolean	false
	structure_timing	true
	isolate_port	disabled

multibit_mode timing_driven

This example sets the **multibit_mode** and **minimum_multibit_width** attributes to their default values.

```
prompt> set_multibit_options -default
```

The following example shows the attributes after the previous command:

```
prompt> report_compile_options
```

```
*****
Report : compile_options
Design : example
Version: J-2014.09-SP5-5
Date  : Tue Nov 3 22:41:16 2015
*****
```

Design	Compile Option	Value
example	flatten	false
	structure	true
	structure_boolean	false
	structure_timing	true
	isolate_port	disabled
	minimum_multibit_width	2
	multibit_mode	non_timing_driven

The following example shows the typical usage flow of the new mbm_timing_opto optimization:

```
prompt> set_multibit_options -mode timing_driven -critical_range default
1
prompt> compile_ultra -spg
prompt> insert_dft
prompt> compile_ultra -spg -incremental
```

SEE ALSO

```
compile_ultra(2)
create_multibit(2)
current_design(2)
remove_attribute(2)
remove_multibit(2)
report_compile_options(2)
report_constraint(2)
report_multibit(2)
reset_design(2)
hdl_infer_multibit(3)
```

set_multicycle_path

Modifies the single-cycle timing relationship of a constrained path.

SYNTAX

```
status set_multicycle_path
    path_multiplier
    [-rise | -fall]
    [-setup | -hold]
    [-start | -end]
    [-from from_list
        | -rise_from rise_from_list
        | -fall_from fall_from_list]
    [-through through_list
        | -rise_through rise_through_list
        | -fall_through fall_through_list]
    [-to to_list
        | -rise_to rise_to_list
        | -fall_to fall_to_list]
    [-reset_path]
    [-comment comment_string]
```

Data Types

```
path_multiplier    integer
from_list          list
rise_from_list     list
fall_from_list     list
through_list       list
rise_through_list  list
fall_through_list  list
to_list            list
rise_to_list       list
fall_to_list       list
comment_string     string
```

ARGUMENTS

path_multiplier

Specifies the number of cycles that the data path must have for setup or hold relative to the startpoint or endpoint clock before data is required at the endpoint. When used with **-setup**, this value is applied to setup path calculations. When used with **-hold**, this value is applied to hold path calculations. If neither **-hold** nor **-setup** are specified, *path_multiplier* is used for setup, and 0 is used for hold. Changing the multiplier for setup also affects the hold check.

-rise

Specifies that rising path delays are affected by *path_multiplier*. The default is that both rising and falling delays are affected. Rise refers to a rising value at the path endpoint.

-fall

Specifies that falling path delays are affected by *path_multiplier*. The default is that both rising and falling delays are affected. Fall refers to a falling value at the path endpoint.

-setup

Specifies that *path_multiplier* is used for setup calculations.

-hold

Specifies that *path_multiplier* is used for hold calculations.

-start | -end

Specifies whether the multicycle information is relative to the period of the start clock or the end clock. These options are only needed for multifrequency designs; otherwise start and end are equivalent. The start clock is the clock source related to the register or primary input at the path startpoint. The end clock is the clock source related to the register or primary output at the path endpoint. The default is to move the setup check relative to the end clock, and the hold check relative to the start clock. A setup multiplier of 2 with **-end** moves the relation forward one cycle of the end clock. A setup multiplier of 2 with **-start** moves the relation backward one cycle of the start clock. A hold multiplier of 1 with **-start** moves the relation forward one cycle of the start clock. A hold multiplier of 1 with **-end** moves the relation backward one cycle of the end clock.

-from from_list

Lists the names of clocks, ports, pins, or cells to use to find path startpoints. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If a cell is specified a cell, one path startpoint on that cell is affected.

-rise_from rise_from_list

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from fall_from_list

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through through_list

Lists the path throughpoints (port, pin, or leaf cell names) of the current design. The multicycle values apply only to paths that pass through one of the points in the *through_list*. If more than one object is included, the objects must be enclosed either in double quotation marks (") or in braces ({}). If you specify the **-through** option multiple times, the multicycle values apply to paths that pass through a member of each *through_list* in the order the lists were given. The path must first pass through a member of the first *through_list*, then through a member of the second list, and so on for every through list specified. If the **-through** option is used in combination with the **-from** or **-to** options, the multicycle values apply only if the **-from** or **-to** conditions are satisfied and the **-through** conditions are satisfied.

-rise_through rise_through_list

Same as the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than once in a single command invocation as with the **-through** option.

-fall_through fall_through_list

Same as the **-through** option, but applies only to paths with a falling transition at the specified objects. You can specify **-fall_through** more than once in a single command invocation as with the **-through** option.

-to *to_list*

Lists the names of clocks, ports, pins or cells to use to find path endpoints. If a clock is specified, all registers and primary outputs related to that clock are used as path endpoints. If a cell is specified, one path endpoint on that cell is affected.

-rise_to *rise_to_list*

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-fall_to *fall_to_list*

Same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-reset_path

Removes existing point-to-point exception information on the specified paths. When used only with **-to**, all paths leading to the specified endpoints are reset. When used only with **-from**, all paths leading from the specified startpoints are reset. When used with **-from** and **-to**, only paths between those points are reset. Only information of the same rise/fall, setup/hold type is reset. This is equivalent to using the **reset_path** command with similar arguments before issuing **set_multicycle_path**.

-comment *comment_string*

Allows the command to accept a comment string. The tool honors the annotation and preserves it with the SDC object so that the exact string is written out when the constraint is written out when you use the **write_sdc** or **write_script** command. The comment remains intact through the synthesis, place-and-route, and timing-analysis flows.

DESCRIPTION

The **set_multicycle_path** command specifies that designated timing paths in the current design have nondefault setup or hold relations.

The synthesis timing engine applies certain rules to determine single-cycle timing relationships for paths between clocked elements. The rules are based on active edges. For flip-flops, a single active edge both launches and captures data. For latches, the open edge is used to launch data, and the close edge is used to latch data.

The setup check ensures that the correct data signal is available on destination registers in time to be properly latched.

The rule for setup is that for multifrequency designs, there can be multiple setup relations between two clocks. For every latch edge of the destination clock, find the nearest launch edge which precedes each capture edge. The smallest difference of (setup_latch_edge - setup_launch_edge) determines the maximum delay requirement for this path.

This is known as single-cycle setup. You can override this default relationship using **set_multicycle_path** or **set_max_delay**. You can apply these commands to clocks, pins, ports, or cells. For example, setting the setup path multiplier to 2 with the **set_multicycle_path** command delays the latch edge one clock pulse. Changing the setup multiplier also affects the default hold check.

The hold check verifies the following items:

- Data from the source clock edge that follows the setup launch edge must not be latched by the setup latch edge.

- Data from the setup launch edge must not be latched by the destination clock edge that precedes the setup latch edge.

The hold check is determined relative to each valid setup relationship, after applying multicycle path multipliers. The most restrictive (largest) difference of (hold_latch_edge - hold_launch_edge) is used as a minimum delay requirement for the path.

The hold relation is conservative. In some cases you might need to override the default hold relation. For multifrequency designs, it is more straightforward to use the **set_min_delay** command to override the default instead of using **set_multicycle_path -hold**.

Setting *path_multiplier* for setup moves the setup check in time by an integer number of active edges. For example, specifying *path_multiplier* of 2 for setup implies a 2 cycle data path.

Most often, the setup check is moved relative to the end clock. This move changes the data latch time at the path endpoint. In multifrequency designs, use the following command to move the data launch time backward:

set_multicycle_path -setup -start

To move the hold relation relative to the end clock use the command

set_multicycle_path -hold -end

If you do not specify **-setup** or **-hold**, the setup relation is set to *path_multiplier* and the hold relation is set to 0.

If you specify **-setup**, the setup multiplier is set to *path_multiplier*. The hold multiplier is unaffected.

If you specify **-hold**, the hold multiplier is set to *path_multiplier*. The setup multiplier is unaffected.

There are separate setup and hold multipliers for rise and fall. In most cases, these are set to the same value. You can use the **-rise** or **-fall** options to apply different values.

The **set_multicycle_path** command is a point-to-point timing exception command. If a path satisfies multiple timing exceptions, the following rules are used in order to determine which exceptions take effect. Rules referring to **-from** apply equally to **-rise_from** and **-fall_from**, and similarly for the rise and fall options of **-through** and **-to**.

- Two **group_path** commands can conflict with each other, but a **group_path** exception alone does not conflict with another type of exception. So the remaining rules apply for two **group_path** exceptions, or two non-**group_path** exceptions.
- If both exceptions are **set_false_paths**, there is no conflict.
- If one exception is a **set_max_delay** and the other is **set_min_delay**, there is no conflict.
- If one exception is a **set_multicycle_path -hold** and the other is **set_multicycle_path -setup**, there is no conflict.
- If one exception is a **set_false_path** and the other is not, the **set_false_path** takes precedence.
- If one exception is a **set_max_delay** and the other is not, the **set_max_delay** takes precedence.
- If one exception is a **set_min_delay** and the other is not, the **set_min_delay** takes precedence.
- If one exception has a -from pin or -from cell and the other does not, the former takes precedence.
- If one exception has a -to pin or -to cell and the other does not, the former takes precedence.
- If one exception has any -through points and the other does not, the former takes precedence.
- If one exception has a -from clock and the other does not, the former takes precedence.
- If one exception has a -to clock and the other does not, the former takes precedence.
- The exception with the more restrictive constraint then takes precedence. For **set_max_delay** and **set_multicycle_path -setup**, this is the constraint with the lower value. For **set_min_delay** and **set_multicycle_path -hold**, it is the constraint with the higher value.

To undo a **set_multicycle_path** command, use **reset_path** or **reset_design**.

Use **set_false_path** to disable setup or hold calculations for paths.

Use **report_timing_requirements** to list the point-to-point exceptions on a design.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example sets all paths between latch1b and latch2d to two-cycle paths for setup. Hold is measured at the previous edge of the clock at latch2d.

```
prompt> set_multicycle_path 2 -from {latch1b} -to {latch2d}
```

The following example moves the hold check to the preceding edge of the start clock:

```
prompt> set_multicycle_path -1 -from {latch1b} -to {latch2d}
```

The following example is a two-phase level-sensitive design, where the designer expects paths from phi1 to phi1 to be zero cycles:

```
prompt> set_multicycle_path 0 -from phi1 -to phi1
```

The following example uses the **-start** option to specify a three-cycle path relative to the clock at the path startpoint. Because clock sources are specified, the constraint affects all sequential elements clocked by that clock and ports with input or output delay relative to that clock.

```
prompt> set_multicycle_path 3 -start -from {clk50mhz} -to {clk10mhz}
```

The following example affects all paths that pass first through the U1 or U2 cell and later pass through the U3 cell. The path resulting in a rise transition at an endpoint is constrained to two cycles, but falling paths are constrained to one cycle.

```
prompt> set_multicycle_path 2 -rise -through {U1 U2} -through {U3}
```

```
prompt> set_multicycle_path 1 -fall -through {U1 U2} -through {U3}
```

The following multifrequency example shows a path from a flip-flop, ff1, clocked by a 20-ns clock to a flip-flop, ff2, clocked by a 10-ns clock, with a multicycle path of two.

```
prompt> create_clock -period 20 -waveform {0 10} clk20
```

```
prompt> create_clock -period 10 -waveform {0 5} clk10
```

```
prompt> set_multicycle_path 2 -setup -from ff1/CP -to ff2/D
```

The single-cycle setup relationship is from the CLK1 edge at 0 ns to the CLK2 edge at 10 ns. With the multicycle path, the setup relationship is now 20 ns (from the CLK1 edge at 0ns to the CLK2 edge at 20ns). The hold relationships are determined according to that setup relationship.

- Data from the source clock edge that follows the setup launch edge must not be latched by the setup latch edge. This implies a hold relationship of 0 ns (from the CLK1 edge at 20 ns to the CLK2 edge at 20 ns).
- Data from the setup launch edge must not be latched by the destination clock edge that precedes the setup latch edge. This implies a hold relationship of 10 ns (from the CLK1 edge at 0 ns to the CLK2 edge at 10 ns).

The most restrictive (largest) hold relationship is used, so the minimum delay requirement for this path is 10 ns. This is a conservative check that might not apply to certain designs. Often you know that the destination register is disabled for the clock edge at 10 ns. You can modify this default hold relationship with the following command to get a 0-ns requirement:

```
prompt> set_min_delay 0 -from ff1/CP -to ff2/D
```

However, an easier approach is to use the following command:

```
prompt> set_multicycle_path 1 -hold -end -from ff1/CP -to ff2/D
```

SEE ALSO

- report_timing_requirements(2)
- reset_design(2)
- reset_path(2)
- set_false_path(2)
- set_max_delay(2)
- set_min_delay(2)

set_mw_lib_reference

Sets the reference library for the Milkyway library.

SYNTAX

```
status_value set_mw_lib_reference  
[-mw_reference_library lib_list]  
[-reference_control_file file_name]  
libName
```

Data Types

```
lib_list    string  
file_name  string  
libName    string
```

ARGUMENTS

-mw_reference_library *lib_list*

Specifies a list of libraries to be set as reference libraries for the Milkyway library.

This argument and **-reference_control_file** are mutually exclusive.

-reference_control_file *file_name*

Specifies a reference control file containing information to set the reference libraries for the Milkyway library.

This argument and **-mw_reference_library** are mutually exclusive.

libName

Specifies the Milkyway library to be worked on.

DESCRIPTION

Sets or changes the reference libraries for the Milkyway library. The Milkyway library being updated must be closed for the command to work correctly. The **-reference_control_file** and **-mw_reference_library** options are mutually exclusive, and at least one of the two must be specified.

A status indicating success or failure is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the reference libraries with a list:

```
prompt> set_mw_lib_reference -mw_reference_library {./lib/ref1 ./lib/ref2} design
```

SEE ALSO

`create_mw_lib(2)`
`set_mw_technology_file(2)`

set_mw_technology_file

Sets the technology file of the Milkyway library.

SYNTAX

```
status set_mw_technology_file  
  [-technology tech_file]  
  [-alf alf_file]  
  libName
```

Data Types

```
tech_file  string  
alf_file   string  
libName    string
```

ARGUMENTS

-technology *tech_file*

Specifies a new technology file to use with the Milkyway library. The command replaces the old technology information completely. Ensure that the new technology information is compatible, or else you must rebuild or re-create the Milkyway library accordingly.

-alf *alf_file*

Specifies a new alf file to use with the Milkyway library. This option is for purpose of importing signal EM data into the library. If there is existing signal EM information in the library, it is completely replaced by the data in the new alf file. Ensure that the new signal EM information is compatible with library technology. Signal EM data in the alf file is updated to the library immediately, but it does not have impact on the opened design.

libName

Specifies the Milkyway library to be updated. The value of *libName* must be a valid Milkyway library name.

DESCRIPTION

This command sets a technology file for a Milkyway library in Milkyway, plib, or alf format. The new information either replaces or is added to the existing library data, depending on the type of information added.

The command returns a status indicating success or failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the technology file of the library design to "new.tf":

```
prompt> set_mw_technology_file -technology new.tf design  
1
```

SEE ALSO

- close_mw_lib(2)
- copy_mw_lib(2)
- create_mw_lib(2)
- current_mw_lib(2)
- open_mw_lib(2)
- rename_mw_lib(2)
- set_mw_lib_reference(2)
- write_mw_lib_files(2)

set_net_routing_layer_constraints

Assigns routing layer constraints to the specified nets.

SYNTAX

```
status set_net_routing_layer_constraints
      list_of_nets
      -min_layer_name minimum_routing_layer_name
      -max_layer_name maximum_routing_layer_name
```

Data Types

<i>list_of_nets</i>	collection
<i>minimum_routing_layer_name</i>	string
<i>maximum_routing_layer_name</i>	string

ARGUMENTS

list_of_nets

Specifies the nets to which the routing layer constraints are applied. This is a required argument.

-min_layer_name *minimum_routing_layer_name*

Specifies the minimum routing layer. You can specify only a single layer name. This is a required option.

-max_layer_name *maximum_routing_layer_name*

Specifies the maximum routing layer. You can specify only a single layer name. This is a required option.

DESCRIPTION

This command assigns routing layer constraints to the specified nets.

The routing layer names you specify must exist in the physical library as well as in the design. In addition, the maximum routing layer must be higher than the minimum routing layer.

If a net already has routing layer constraints set, the tool issues a warning message and overrides the existing routing layer constraints.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example assigns routing layer constraints to the net named netA.

```
prompt> set_net_routing_layer_constraints {netA}  
-min_layer_name metal1 -max_layer_name metal3
```

SEE ALSO

```
report_net_routing_layer_constraints(2)  
remove_net_routing_layer_constraints(2)  
set_route_zrt_common_options(2)
```

set_net_routing_rule

Assigns a nondefault or default routing rule to specific nets.

SYNTAX

```
status set_net_routing_rule  
  -rule rule_name  
  list_of_nets  
  [-reroute normal | minorchange | freeze]  
  [-timing_driven_spacing]  
  [-top_layer_probe AnyPort | OutPort | AllPort]  
  [-rule_is_user one-of-string]
```

Data Types

```
rule_name      string  
list_of_nets  list
```

ARGUMENTS

-rule *rule_name*

Specifies the name of the routing rule to be assigned.

The *rule_name* argument can be either the name of an existing nondefault routing rule or the **default** keyword, which assigns the default routing rule to the specified nets.

list_of_nets

Specifies the nets to which to assign the routing rule.

-reroute normal | minorchange | freeze

Specifies the rerouting mode.

Valid values are

- **normal** (the default)
Performs a typical reroute on the net.
- **minorchange**
Reroutes only the minor changes to the nets.
- **freeze**
Does not perform any rerouting.

This option is not supported in Design Compiler topographical mode.

-timing_driven_spacing

Assigns timing-driven spacing constraints to the specified nets.

When crosstalk problems occur around the specified nets or the intralayer coupling capacitance of the nets is too high, the router spaces this net away from other nets and avoids placing long parallel wires next to this net.

The classic router honors this constraint during global routing and detail routing. Zroute honors this constraint only during global routing; during detail routing, this constraint is ignored.

This option is not supported in Design Compiler topographical mode.

-top_layer_probe AnyPort | OutPort | AllPort

Specifies how to route the nets close to ports at the top of the routing layers so that the top-layer segment of the net can be used for probing.

Valid values are

- **AnyPort** (the default)
Routes the net from either an input port or output port of the net at the top of the routing layers. The net needs to reach the top layer only once; the location does not matter. This is for signal probing only.
- **OutPort**
Routes only the net from an output port of the net at the top of the routing layers. The routing from the output pin has to reach the top layer prior to hitting the Steiner point or another pin. This is for post-mask surgery, which might want to disconnect the driving pin of the net.
- **AllPort**
Routes the net from the input and output ports of the net at the top of the routing layers. The routing from every pin has to reach the top layer prior to hitting the Steiner point or another pin. This is for post-mask surgery, which might want to disconnect any pin from the net.

For two-pin nets, all three options are essentially the same. You will only notice a difference for nets with three or more pins.

This option is for the classic router only.

This option is not supported in Design Compiler topographical mode.

DESCRIPTION

This command assigns a nondefault or default routing rule to specific nets. Nondefault rules are defined in the physical library or by using the **define_routing_rule** command and can have associated user-defined width and spacing rules and via types. These rules govern nondefault wiring such as the wide wires and vias that are typically used for clock signals and other sensitive signals.

Library-specific rules are defined in the physical library and stored in the .pdb file.

If nondefault routing rules are not defined in the physical library provided with the design database, you can define them in a separate .plib file that can be updated into the existing .pdb file by using the **update_lib** command. For more information about the **update_lib** command, see the man page.

You can define design-specific rules by using the **define_routing_rule** command and storing them in the design .db file.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example assigns a nondefault rule named WideMetal to a net named B_CLK.

```
prompt> set_net_routing_rule {B_CLK} -rule WideMetal
```

SEE ALSO

update_lib(2)
define_routing_rule(2)
report_routing_rules(2)

set_net_search_pattern_delay_estimation_options

Sets delay estimation options for a pattern.

SYNTAX

```
status set_net_search_pattern_delay_estimation_options  
-pattern id  
[-default]  
[-rule rule_name]  
[-min_layer_name layer_name]  
[-max_layer_name layer_name]
```

Data Types

```
id          integer  
rule_name   string  
layer_name  string
```

ARGUMENTS

-pattern *id*

This required option specifies the pattern ID that the delay estimation options will be applied to. The corresponding search pattern should already be defined.

-default

Resets all options to default, meaning that no rules or layer assignments will be applied to the nets that match the specified pattern ID. The **-default** option cannot be used with any other options.

-rule *rule_name*

Assigns a non-default routing rule to a specified net pattern ID. Once defined, the unit Rs and unit Cs of the matching nets will change accordingly.

This option is not supported in Design Compiler topographical mode.

-min_layer_name *layer_name*

Defines a minimum routing layer for nets that match the specified pattern ID. It must be used with the **-max_layer_name** option.

-max_layer_name *layer_name*

Defines a maximum routing layer for nets that match the specified pattern ID. It must be used with the **-min_layer_name** option.

DESCRIPTION

The **set_net_search_pattern_delay_estimation_options** command specifies layer assignments, which will ultimately alter the RC values of the matching nets. For example, if min and max routing layers of M9 and M10 are specified, virtual route-based optimizations will use the average R and C of layers M9 and M10 to compute RC delays for all matched nets.

In order for the layer constraints to be honored by the routers, net search patterns and corresponding delay estimation options must be applied before you run the **compile_ultra -spg -layer_optimization** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how **set_net_search_pattern_delay_estimation_options** can be used to assign a min routing layer of METAL7 and a max routing layer of METAL8 to all nets that match pattern ID 1.

```
prompt> set_net_search_pattern_delay_estimation_options -pattern 1 -min_layer_name METAL7 -max_layer_name METAL8
```

SEE ALSO

```
create_net_search_pattern(2)
report_net_search_pattern(2)
remove_net_search_pattern(2)
report_net_search_pattern_delay_estimation_options(2)
set_net_search_pattern_priority(2)
report_net_search_pattern_priority(2)
get_matching_nets_for_pattern(2)
```

set_net_search_pattern_priority

Sets the net search pattern matching priority.

SYNTAX

```
status set_net_search_pattern_priority  
      string | -default
```

ARGUMENTS

string

Provides a list of IDs of net patterns that have already been created.

-default

Resets the prioritization of net pattern matching. When you use the **-default** option, net pattern matching prioritization reverts to the original behavior, which is to use the order in which patterns were created. For example, if the **create_net_search_pattern** command was used only three times, the default matching priority will be "1 2 3".

DESCRIPTION

When multiple net patterns are defined, a net can match more than one search pattern. For example, if you create a search pattern for nets with a length less than 3 and another search pattern for nets with a length between 2 and 4, then a net with a length of 2.5 will match both search patterns.

During optimization, however, the first net pattern that was created takes precedence over the second net pattern. Therefore, in the previous example, the net with a length of 2.5 will belong to the first collection by default. If you want to change this behavior during optimization, you can specify **set_net_search_pattern_priority "2 1"**, and then the net with a length of 2.5 will belong to the second collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how the **set_net_search_pattern_priority** command can be used to change the priority of five existing patterns, which would by default have a match priority of "1 2 3 4 5".

```
prompt> set_net_search_pattern_priority "4 5 1 2 3"
```

SEE ALSO

- create_net_search_pattern(2)
- report_net_search_pattern(2)
- remove_net_search_pattern(2)
- report_net_search_pattern_delay_estimation_options(2)
- report_net_search_pattern_priority(2)
- get_matching_nets_for_pattern(2)

set_opcond_inference

Specifies the strategy for operating condition inference.

SYNTAX

```
status set_opcond_inference
  [-level level_value]
  [-match_process_temperature true | false]
  [-object_list cells]
  [-applies_to macro_cells pad_cells switch_cells]
```

Data Types

level_value string
cells collection

ARGUMENTS

-level *level_value*

Specifies the degree of freedom for the tool to infer the operating conditions. There are four possible levels of operating condition inference:

- **exact** The timing on the cells is critical and the tool is not allowed to infer operating conditions.
- **unique_resolved** In case of potential **LIBSETUP-001** errors, the tool infers an operating condition on the cells when the library contains only one cell that matches the reference name of the cell. If there are multiple library cells with the same reference name, the tool does not infer the operating condition on the cell.
- **closest_resolved** When the library contains multiple library cells that match the reference name, the tool attempts to select the cell whose operating condition is closest to the constrained operating condition. The tool requires exactly one library cell that is closest to the constrained operating condition than any other library cell. If there are multiple library cells that are equally close to the operating condition, the tool does not infer the operating condition.
- **closest_unresolved** When you use this option, the tool has maximum flexibility when inferring the operating conditions. The requirement to have a single library cell closer to the operating condition, than any other library cell is also waived.

When you do not use any of these options, the default is **closest_resolved**.

For more information about how the closest library cell relative to the operating condition is defined, see the user guide.

-match_process_temperature true | false

When this option is set to true, library cells whose process and temperature do not match the process and temperature in the specified operating condition, are ignored while inferring the operating condition. This option has no effect when the value of the **-level** option is **exact**. The default is **false**.

-object_list cells

Specifies the cells to which the operating condition inference strategy applies. The cells must be macro, pad, switch, or hierarchical cell.

When you do not specify the cells, the settings are applied to the top design. When hierarchical cells are specified, the strategy is applied to the instances and their child instances. When the **-applies_to** option is also specified, only child cells of the specified cell types are included.

-applies_to macro_cells pad_cells switch_cells

Specifies the cell types to which the strategy is applied. If this option is not specified, macro cells, pad cells and switch cell types are considered for the operating condition inference.

DESCRIPTION

If the specified operating condition does not match any available cell in the library and the timing on the cell is not critical, the tool can infer the operating condition on the macro, pad, and switch cells to avoid potential **LIBSETUP-001** errors.

The command specifies how the tool should infer operating conditions on macro, pad, and switch cells. At least one of **-level** or **-match_process_temperature** options must be specified. The command applies to the identified instances and their child instances. A strategy defined at a lower level that applies to the same cell type overrides the strategy specified at a higher level.

Multicorner-Multimode Support

This command applies to all scenarios.

EXAMPLES

The following example specifies that operating condition inference should not be performed on all macro cells under the hierarchical cell **subA**.

```
prompt> set_opcond_inference -level exact -object_list [get_cells {subA}] -applies_to macro_cells
```

SEE ALSO

report_opcond_inference(2)
set_operating_conditions(2)
set_voltage(2)
report_lib(2)

set_operating_conditions

Defines the operating conditions for the current design.

SYNTAX

```
status set_operating_conditions  
  [-analysis_type bc_wc | on_chip_variation]  
  [-min min_condition]  
  [-max max_condition]  
  [-min_library min_lib]  
  [-max_library max_lib]  
  [-min_phys min_proc]  
  [-max_phys max_proc]  
  [-library lib]  
  [-object_list objects]  
  [condition]
```

Data Types

<i>min_condition</i>	list
<i>max_condition</i>	list
<i>objects</i>	list
<i>condition</i>	list

ARGUMENTS

-analysis_type bc_wc | on_chip_variation

Specifies how to use the operating conditions. The **bc_wc** and **on_chip_variation** options are mutually exclusive; use only one per command.

Specifying either **bc_wc** or **on_chip_variation** switches the design to min_max mode. The **bc_wc** value specifies that the min and max operating conditions are two extreme operating conditions. In the **bc_wc** analysis, setup violations are checked only for the maximum operating condition, and the hold violations are checked only for the minimum operating condition.

The **on_chip_variation** value specifies that the minimum and maximum operating conditions represent, respectively, the lower and upper bounds of the maximum variation of operating conditions on the chip. All maximum path delays use the maximum operating condition, and all minimum path delays use the minimum operating condition.

-min *min_condition*

Specifies the operating condition to use for minimum delay analysis. If you do not specify an operating condition for minimum delay analysis, the tool uses the maximum operating condition. The **-min** option must be used with the **-max** option.

-max *max_condition*

Specifies the operating condition to use for maximum delay analysis.

-min_library *min_lib*

Specifies the library containing definitions of the operating conditions for minimum delay analysis. This is either a library name or a collection. The tool selects the first library in the collection containing the definitions.

-max_library *max_lib*

Specifies the library containing definitions of the operating conditions for maximum delay analysis. This is either a library name or a collection. The tool selects the first library in the collection containing the definitions.

-min_phys *min_proc*

Specifies the name of the process resource to search for the resistance and capacitance values for minimum delay analysis. This option must be used in with the **-max_phys** option.

-max_phys *max_proc*

Specifies the name of the process resource to search for the resistance and capacitance values for maximum delay analysis. This argument must be used with the **-min_phys** option.

-library *lib*

Specifies the library containing definitions of the operating conditions for both maximum and minimum delay analysis. This is either a library name or a collection. The tool selects the first library in the collection containing the definitions.

-object_list *objects*

Specifies the cells or ports on which to set operating conditions. If you do not use this option, operating conditions are set on the design. This option accepts both leaf cells and hierarchical blocks. This option is only available for multivoltage features and requires the appropriate license.

condition

Specifies conditions that define environmental characteristics to use during maximum and minimum delay analysis.

DESCRIPTION

This command defines the operating conditions (or environmental characteristics) under which to time or optimize the current design. The operating conditions specified must be defined in *lib* or in one of the libraries in the **link_library**. A **local_link_library** set on the current design is added to the beginning of the **link_library**, before the **link_library** is searched. The order for a library search is as follows:

1. *lib*
2. **local_link_library**
3. **link_library**

If you do not specify any operating conditions for a design, the **compile** command searches in the first library of the link library for default operating conditions. If the library does not have default operating conditions, no operating conditions are used.

Operating conditions set by using the **-object_list** option override the operating conditions set on design or higher levels of hierarchy.

To see the operating conditions that are defined for the current design, and to see which libraries the current design is linked to, use the **report_design** command. To see the operating conditions defined in the specified library, use the **report_lib** command.

To remove operating conditions from the current design, use **set_operating_conditions** without specifying any operating conditions, or use the **reset_design** command.

Multicorner-Multimode Support

This command applies to the current scenario only.
You must use the **set_operating_conditions** command to define the operating conditions for each scenario.

EXAMPLES

The following example shows an operating condition definition, as it appears in the source text of a library:

```
operating_conditions("BCCOM") {
  process : 0.6 ;
  temperature : 20 ;
  voltage : 5.25 ;
  tree_type : "best_case_tree" ;
}
```

The name of this set of operating conditions is *BCCOM*. The parameters are defined as follows:

process

A floating-point number that represents the characteristics of a semiconductor manufacturing process.

temperature

A floating-point number that represents the temperature of the defined environment.

voltage

A floating-point number that defines the upper boundary of the voltage range in which the defined environment operates. The lower boundary is always 0.0.

tree_type

The interconnect model for the environment. The **compile** command uses the interconnect model to select a formula for calculating interconnect delays. Three models are available:

- *best_case_tree*, which assumes the net delay to be 0
- *worst_case_tree*, which uses the lumped RC model
- *balanced_tree*, in which all loads share the wire resistance evenly.

When process factor, operating temperature, and operating voltage deviate from their nominal values, **compile** uses a linear model to compensate for the effect of deviations on such values as cell delays, input loads, and output drives. The nominal values used are defined in the same library that contains the definitions of the set of operating conditions.

The following example sets the operating conditions to *WCIND* if the link_library is *my_lib.db*, and the design does not have a local_link_library set. The library name for *my_lib.db* is *my_lib_core*.

```
prompt> set_operating_conditions WCIND
```

Using operating condition 'WCIND' found in library "my_lib_core".

In the following example, the BCIND values found in *other_lib.db* are used for minimum delay analysis, and WCIND values are used for maximum delay analysis. The library name for *other_lib.db* is *other_lib_core*.

```
prompt> set_operating_conditions -min BCIND -max WCIND \  
       -library other_lib_core
```

Using operating condition 'BCIND' found in library 'other_lib_core'.

Using operating condition 'WCIND' found in library 'other_lib_core'.

The following example shows how to remove operating conditions defined for the current design:

```
prompt> set_operating_conditions
```

SEE ALSO

- compile(2)
- report_lib(2)
- reset_design(2)
- set_local_link_library(2)
- link_library(3)

set_opposite

Defines two input ports as logically opposite.

SYNTAX

```
int set_opposite  
    port1  
    port2
```

Data Types

```
port1  string  
port2  string
```

ARGUMENTS

port1 port1

Input port names in the current design that are logically opposite.

DESCRIPTION

Defines two input ports in the current design as logically opposite. Use the command to eliminate redundant inverters and improve the quality of optimization. Inconsistencies in logical relations among ports are detected. It is an error to specify such an inconsistency using **set_opposite**.

Use the **reset_design** command to remove this property from a port.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets two input ports named "A" and "B" to be logically opposite.

```
prompt> set_opposite A B
```

SEE ALSO

reset_design(2)
set_equal(2)

set_optimize_dft_options

Defines options for physical design-for-test (DFT) optimization.

SYNTAX

```
status set_optimize_dft_options  
  [-repartitioning_method none | single_directional | multi_directional | adaptive]  
  [-single_dir_option horizontal | vertical]
```

ARGUMENTS

-repartitioning_method none | single_directional | multi_directional | adaptive

Specifies the type of the scan chain repartition algorithm used in placement-aware scan optimization.

This command affects DFT optimization performed by the **compile_ultra -scan -incremental** command performed after DFT insertion.

If you specify **none**, DFT optimization does not do scan repartitioning.

If you specify **single_directional**, DFT optimization uses the traditional single-directional scan repartitioning algorithm. By default, the direction is **horizontal**. You can use the **-single_dir_option** option to specify the desired direction.

If you specify **multi_directional**, DFT optimization uses the multi-directional scan repartitioning algorithm. The multi-directional algorithm is designed to achieve better scan path total wire-length reduction efficiency, especially for designs with a large number of scan chains.

If you specify **adaptive**, DFT optimization chooses the scan repartitioning algorithm adaptively, according to the number of scan chains, for the best scan wire-length reduction efficiency. For designs with no more than 20 scan chains, the traditional single-directional scan repartitioning algorithm is used; otherwise, the multi-directional scan repartitioning algorithm is used. If the single-directional algorithm is chosen, the partition direction is determined by the setting of the **-single_dir_option** option.

The default of this option is **multi_directional**.

-single_dir_option horizontal | vertical

Specifies the desired direction for the single-directional scan repartitioning algorithm. The option takes effect only when the **-repartitioning_method** is set to **single_directional** or set to **adaptive** on a design with 20 or fewer scan chains.

The default of this option is **horizontal**.

DESCRIPTION

This command defines the options that are used for DFT optimization.

SEE ALSO

`compile_ultra(2)`
`remove_optimize_dft_options(2)`
`report_optimize_dft_options(2)`

set_optimize_registers

Sets the **optimize_registers** attribute on the specified designs or on the current design, so that **compile** automatically invokes the DC Ultra **optimize_registers** command to retime the design during optimization.

SYNTAX

```
status set_optimize_registers
[true | false]
[-designs design_list]
[-minimum_period_only]
[-sync_transform multiclass | decompose | dont_retime]
[-async_transform multiclass | decompose | dont_retime]
[-check_design [-verbose]]
[-print_critical_loop]
[-clock clock_name [-edge rise | fall]]
[-latch]
[-justification_effort low | medium | high]
[-delay_threshold target_clock_period]
```

Data Types

<i>design_list</i>	list
<i>clock_name</i>	string

ARGUMENTS

true | false

Specifies the value with which to set the **optimize_registers** attribute. The default value is **true**.

-designs *design_list*

Specifies a list of designs to retime. The default is the current design.

-minimum_period_only

Indicates that only the minimum period step of the retiming algorithm (minimum clock period retiming) and not the minimum area (sequential cell count optimization) step is to be executed during compile. By default, both the minimum period and minimum area optimization steps are executed. This option is useful if you want a fast turnaround when trying to optimize the design's timing. The runtime is reduced, but your area results will not be optimal. After you are satisfied with the timing, you can attempt to reduce area by running the retiming command without this option.

-sync_transform multiclass | decompose | dont_retime

Specifies which transformation method is used for synchronous sequential cells in the design. An edge-triggered register is

synchronous if none of its input pins change the outputs asynchronously. A level-sensitive latch is considered synchronous if none of its input pins can change the outputs during the clock phase where the latch is not transparent.

Selecting **multiclass** transformation specifies that the identifiable synchronous clear, set, and enable functionality is moved with the synchronous sequential cells if they are moved during retiming. Sequential cells are classified according to their set, clear, and enable connections. The class of the sequential cells at the fanin or fanout of a combinational cell determines whether retiming across this cell can be performed.

Selecting the **decompose** transformation specifies that synchronous sequential cells in the design are transformed into an instance of a D-flip-flop or D-latch and additional combinational logic to create the necessary synchronous functionality. Only the D-flip-flop/D-latch instance can be moved during retiming.

Specifying **dont_retime** specifies that synchronous sequential cells will not be moved during retiming. Their mapping might still be changed to a different flip-flop from the technology library. The **dont_touch** attributes and retiming transformation attributes set on individual sequential cells override the value set in this option. To set retiming transform attributes, use the **set_transform_for_retiming** command. The default argument for this option is *multiclass*.

-async_transform multiclass | decompose | dont_retime

Specifies which transformation method is used for asynchronous sequential cells in the design. An edge-triggered register is asynchronous if at least one of its input pins changes the outputs asynchronously. A level-sensitive latch is asynchronous if at least one of its inputs can change the outputs during the clock phase where the latch is not transparent.

Selecting the **multiclass** transformation specifies that the identifiable asynchronous clear and set functionality, as well as any synchronous set, clear, and enable functionality, is moved with the asynchronous sequential cells, if they are moved during retiming. Sequential cells are classified according to their set, clear, and enable connections. The class of the sequential cells at the fanin or fanout of a combinational cell determines whether retiming can be performed across this cell.

Selecting the **decompose** transformation specifies that asynchronous sequential cells in the design are transformed into an instance of a flip-flop or latch with asynchronous set and clear inputs, as necessary, and additional combinational logic to create the necessary synchronous functionality. Only the flip-flop/latch instances can be moved during retiming. They will be classified according to their synchronous set, clear, and enable functionality.

Selecting the **dont_retime** value specifies that asynchronous sequential cells will not be moved during retiming. Their mapping might still be changed to a different flip-flop from the technology library. The **dont_touch** attributes and retiming transformation attributes set on individual sequential cells override the value set in this option. To set retiming transform attributes, use the **set_transform_for_retiming** command. The default value for this option is *multiclass*.

-check_design

Indicates that additional information about the design is to be displayed before and after retiming. This information includes the number of cells in different categories (for example, hierarchy cells with **dont_touch** attributes or non-movable sequential cells) and more detailed information about the selection of the preferred flip-flop or latch. Use this information to help in troubleshooting if retiming does not show the expected results.

-verbose

Indicates that the explicit names of the cells are to be displayed along with the number of cells in each category for most categories of the **-check_design** option. The explicit naming of cells can help to locate a problem; however, the lists of output names might be long. This option can be used only with the **-check_design** option.

-print_critical_loop

Indicates that the critical loop of the design, as seen during retiming, is to be displayed. The critical loop is defined as the sequence of directly-connected combinational and sequential cells whose total combinational delay divided by the number of registers in the loop has a higher value than any other loop in the design. The critical loop limits the minimum clock period that can be achieved by retiming. Use this option to help troubleshoot problem areas of the design if the intended clock period cannot be achieved with the given number of sequential cells in the design. If you are pipelining a data path, you might need to add pipeline stages in the HDL code.

-clock *clock_name*

Specifies the name of the clock whose sequential cells are to be retimed. The clock must not be a virtual clock; it must have an associated clock port. The name of the clock is either the name specified in the **create_clock** command or the name of the clock port, if the **create_clock** command had no name specified. The registers of the clock are all sequential cells that are triggered by this clock. The connection from the clock port to the sequential cell can be through clock gating cells, buffer cells and inverter cells. If the **-clock** option is specified and edge-triggered registers are retimed, registers of other clocks are not retimed. If level-sensitive latches are retimed and the **-clock** option is specified, the latches driven by this clock as well as those driven by other clocks needed to complete a two-phase clock system are retimed. If edge-triggered registers are retimed, only registers triggered by one specific edge of the clock are retimed. By default the registers triggered by the rising edge are retimed. A different edge can be specified using the **-edge** option.

-edge *rise* | *fall*

Specifies whether the registers triggered by the rising or the falling edge of the clock are to be retimed. This option can only be used together with the **-clock** option. When level-sensitive latches are retimed this option does not matter.

-latch

Specifies that level-sensitive latches are to be retimed instead of edge-triggered sequential cells (flip-flops). If this option is used the edge-triggered sequential cells in the design will not be moved. In order to be able to retime latches they must be driven by a symmetrical two-phase clocks system. Latches that are used to prevent glitches in gated clocks will not be moved, even if the **-latch** option is used. These latches are in the fanin of clock-gating cells.

-justification_effort *low* | *medium* | *high*

Specifies the justification effort level that is to be used during backward justification of registers. You can specify a value of low, medium, or high. Specifying low ensures that justification terminates very quickly, but the QoR may be bad. Specifying medium provides good QoR with considerable runtime. Specifying high provides the best QoR without any regard for runtime. The default value for this option is medium.

-delay_threshold *target_clock_period*

Instructs *Design Compiler* to improve paths with delays less than worst delay as given in critical loop report. Design Compiler makes additional retiming passes until all such paths are optimized to operate at given target clock period.

DESCRIPTION

This command sets the **optimize_registers** attribute on the specified design or on the current design, so that the **compile** command automatically invokes **optimize_registers** to retime the design during optimization. If you specify any of the new options, then **compile** invokes **optimize_registers** with the new options. The DC Ultra **optimize_registers** command moves registers in a design across combinational logic to achieve a target clock period, then minimizes the registers while maintaining that clock period. The **optimize_registers** attribute is particularly useful for creating embedded dc_shell compilation scripts for HDL descriptions that are to be transformed to DesignWare synthetic library components.

For more information about retiming during optimization, see the Design Compiler documentation.

To remove **optimize_registers** use the **remove_attribute** command. You can achieve the same effect by setting the **optimize_registers** attribute to false; execute **set_optimize_registers false**. The **reset_design** command removes all attributes, including **optimize_registers**.

Licensing during compile when using the **optimize_registers** attribute can be controlled by using the **compile_retime_license_behavior** dc_shell variable.

The **optimize_registers** attribute can also be used to direct Design Compiler to optimize subcritical paths using delay threshold optimization. Subcritical paths are paths that operate at delays less than the final worst critical delay provided by retiming. Threshold

optimization techniques can be used to balance delays on paths with positive slacks. The **optimize_registers** attribute uses the target clock period specified by **-delay_threshold** and performs multiple passes of min-period retiming on paths that operate at delays less than the worst delay but greater than the target clock period.

EXAMPLES

In the following example, **optimize_registers** is enabled on the TEST design:

```
prompt> set_optimize_registers -design TEST
```

In the following example, **optimize_registers** is enabled on the current design and disabled on the OLD design:

```
prompt> read TEST
prompt> set_optimize_registers
prompt> set_optimize_registers false -design OLD
```

In the following example, **optimize_registers** is enabled on the TEST design with the **-minimum_period_only** option. This means that when compile automatically calls **optimize_registers** under the hood, it invokes **optimize_registers** with **-minimum_period_only**.

```
prompt> set_optimize_registers -design TEST -minimum_period_only
```

SEE ALSO

- compile(2)
- current_design(2)
- optimize_registers(2)
- remove_attribute(2)
- reset_design(2)
- attributes(3)

set_output_clock_port_type

Specifies the output clock port as a data port or a clock port.

SYNTAX

```
status set_output_clock_port_type  
-data | -clock  
port_list
```

Data Types

port_list list

ARGUMENTS

-data

Specifies the port type as data. You must specify either the **-data** or the **-clock** option; they are mutually exclusive. The default type is data.

-clock

Specifies the port type as clock. You must specify either the **-data** or **-clock** option; they are mutually exclusive.

port_list

Specifies the ports to which the port type is assigned.

DESCRIPTION

This command specifies the type of output port.

An output clock port is the port that the clock can reach. An output clock port can drive either clock pins or data pins. The tool cannot determine the type of pin to which the output clock port will be connected. You must specify either **-data** or **-clock** as the port type. By default, all output clock ports will be treated as data ports.

Note that even if the port type is set to clock, the output delay will still be honored, which may impact the timing result. For this reason, it is considered best practice not to set output delay on an output clock port, if the port has already been set as clock port.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example specifies the port named OUT1 as a clock port:

```
prompt> set_output_clock_port_type -clock OUT1
```

set_output_delay

Sets output delay on pins or output ports relative to a clock signal.

SYNTAX

```
status set_output_delay
    delay_value
    [-reference_pin pin_port_name]
    [-clock clock_name [-clock_fall] [-level_sensitive]]
    [-network_latency_included]
    [-source_latency_included]
    [-rise]
    [-fall]
    [-max]
    [-min]
    [-add_delay]
    [-group_path group_name]
    port_pin_list
```

Data Types

<i>delay_value</i>	float
<i>clock_name</i>	string or collection
<i>group_name</i>	string
<i>port_pin_list</i>	list

ARGUMENTS

delay_value

Specifies the path delay. The *delay_value* must be in units consistent with the technology library used during optimization. The *delay_value* represents the amount of time that the signal is required before a clock edge. For maximum output delay, this usually represents a combinational path delay to a register plus the library setup time of that register. For minimum output delay, this value is usually the shortest path delay to a register minus the library hold time.

-reference_pin *pin_port_name*

Specifies the clock pin or port to which the specified delay is related. If you use this option, and if propagated clocking is being used, the delay value is related to the arrival time at the specified reference pin, which is clock source latency plus its network latency from the clock source to this reference pin. The options *-network_latency_included* and *-source_latency_included* cannot be used at the same time as the *-reference_pin* option. For ideal clock network, only source latency is applied.

The pin specified with the *-reference_pin* option should be a leaf pin or port in a clock network, in the direct or transitive fanout of a clock source specified with the *-clock* option. If multiple clocks reach the port or pin where you are setting the input delay, and if the *-clock* option is not used, the command considers all of the clocks.

-clock *clock_name*

Specifies the clock to which the specified delay is related. If **-clock_fall** is used, **-clock *clock_name*** must be specified. If **-clock** is not specified, the delay is relative to time zero for combinational designs. For sequential designs, the delay is considered relative to a new clock with period determined by considering the sequential cells in the transitive fanout of each port.

The clock can be either a string or collection of one object.

-clock_fall

Specifies that the delay is relative to the falling edge of the clock. If **-clock** is specified, the default is the rising edge.

-level_sensitive

Specifies that the destination of the delay is a level-sensitive latch. This allows the tool to derive setup and hold relationship for paths to this port as if it were a level-sensitive latch. If **-level_sensitive** is not used, the output delay is treated as if it were a path to a flip-flop.

-network_latency_included

Specifies that the clock network latency should not be added to the output delay value. If this option is not specified, the clock network latency of the related clock is added to the output delay value. It has no effect if the clock is propagated or the output delay is not specified with respect to any clock.

-source_latency_included

Specifies that the clock source latency should not be added to the output delay value. If this option is not specified, the clock source latency of the related clock is added to the output delay value. It has no effect if the output delay is not specified with respect to any clock.

-rise

Specifies that *delay_value* refers to a rising transition on specified ports of the current design. If neither **-rise** nor **-fall** is specified, then rising and falling delays are assumed to be equal.

-fall

Specifies that *delay_value* refers to a falling transition on specified ports of the current design. If neither **-rise** nor **-fall** is specified, then rising and falling delays are assumed to be equal.

-max

Specifies that *delay_value* refers to the longest path. If neither **-max** nor **-min** is specified, maximum and minimum output delays are assumed to be equal.

-min

Specifies that *delay_value* refers to the shortest path. If neither **-max** nor **-min** is specified, maximum and minimum output delays are assumed to be equal.

-add_delay

Specifies whether to add delay information to the existing output delay, or to overwrite. The **-add_delay** option enables you to capture information about multiple paths leading from an output port that are relative to different clocks or clock edges.

For example, the following command removes all other maximum rise output delay from **OUT1**, since **-add_delay** is not specified. Other output delay with a different clock or with **clock_fall** is removed.

```
prompt> set_output_delay 5.0 -max -rise -clock phi1 {OUT1}
```

In the following example, **-add_delay** is specified:

```
prompt> set_output_delay 5.0 -max -rise -clock phi1 \
      -add_delay {Z}
```

If there is an output maximum rise delay for **Z** relative to the clock **phi1** rising edge, the larger value is used. The smaller value does not result in critical timing for maximum delay. For minimum delay, the smaller value is used. If there is maximum rise output delay relative to a different clock or different edge of the same clock, it remains with the new delay.

-group_path *group_name*

Specifies that paths ending at the specified ports or pins are added into the named group. If the group does not already exist, it is created. This is equivalent to specifying the following command in addition to **set_output_delay**:

```
prompt> group_path -name group_name -to port_pin_list
```

If **-group_path** is not specified, the existing path grouping is not changed.

port_pin_list

A list of output port or internal pin names in the current design to which *delay_value* is assigned. If more than one object is specified, the objects are enclosed in double quotation marks (") or in braces ({}). If output delay is specified on a pin, the cell of the pin is set to size only to leave room for compile applying sizing on it.

DESCRIPTION

This command sets output path delay values for the current design. Used with the **set_load** and **set_driving_cell** commands, the input and output delays characterize the operating environment of the current design.

The **set_output_delay** command sets output path delays on output ports relative to a clock edge. Output ports are assumed to have no output delay unless specified. For inout (bidirectional) ports, you can specify the path delays for both input and output modes.

To describe a path delay to a level-sensitive latch, use the **-level_sensitive** option. If the latch is positive-enabled, set the output delay relative to the rising clock edge. If the latch is negative-enabled, set the output delay relative to the falling clock edge. If time is being borrowed at that latch, subtract that time borrowed from the path delay to the latch when determining output delay.

The **characterize** command automatically sets input and output delay, drive, and load values based on the environment of a cell instance.

The tool adds input delay to path delay for paths starting at primary inputs and output delay for paths ending at primary outputs.

The **-group_path** option modifies the path grouping. Path grouping affects the maximum delay cost function. The worst violator within each group adds to the cost. For optimization, grouping some paths separately may improve their delay cost, but it may also increase design area and compile time.

Use **report_port** to list output delays associated with ports. To list output delays of internal pins, use **report_design**. Use **report_path_group** to list the path groups which are defined.

Use **remove_output_delay** or **reset_design** to remove output delay values. To modify paths grouped with the **-group_path** option, use the **group_path** command to place the paths in another group or the default group.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example sets an output delay of 1.7 relative to the rising edge of CLK1 for all output ports in the design.

```
prompt> set_output_delay 1.7 -clock [get_clocks CLK1] \
[all_outputs]
```

The following example sets the input and output delays for the INOUT1 bidirectional port. The input signal arrives at INOUT1 2.5 units after the falling edge of CLK1. The output signal is required at INOUT1 at 1.4 units before the rising edge of CLK2.

```
prompt> set_input_delay 2.5 -clock CLK1 -clock_fall {INOUT1}
prompt> set_output_delay 1.4 -clock CLK2 {INOUT1}
```

In the following example, there are three paths from the OUT1 output port. One path is relative to the rising edge of CLK1. Another path is relative to the falling edge of CLK1. The third path is relative to the falling edge of CLK2. The **-add_delay** option is used to indicate that new output delay information will not cause old information to be removed.

```
prompt> set_output_delay 2.2 -max -clock CLK1 \
-add_delay {OUT1}
prompt> set_output_delay 1.7 -max -clock CLK1 \
-clock_fall -add_delay {OUT1}
prompt> set_output_delay 4.3 -max -clock CLK2 \
-clock_fall -add_delay {OUT1}
```

The following example specifies two maximum delays and two minimum delays for the Z port using the **-add_delay** option. Since the information is relative to the same clock and clock edge, only the largest of the maximum values (5.0) and the smallest of the minimum values (1.1) are maintained. If the **-add_delay** option is not used, the new information overwrites the old information.

```
prompt> set_output_delay 3.4 -max -clock CLK1 \
-add_delay {Z}
prompt> set_output_delay 5.0 -max -clock CLK1 \
-add_delay {Z}
prompt> set_output_delay 1.1 -min -clock CLK1 \
-add_delay {Z}
prompt> set_output_delay 1.3 -min -clock CLK1 \
-add_delay {Z}
```

The following example uses the **-group_path** option to add ports into a named group. Without this option, paths to these ports are included in the CLK group.

```
prompt> set_output_delay 4.5 -max -clock CLK \
-group_path busA {busA[*]}
```

SEE ALSO

- all_outputs(2)
- characterize(2)
- create_clock(2)
- group_path(2)
- remove_output_delay(2)
- report_design(2)

```
report_path_group(2)
report_port(2)
reset_design(2)
set_driving_cell(2)
set_load(2)
set_max_delay(2)
set_output_delay(2)
```

set_partial_on_translation

Defines the translation of PARTIAL_ON to FULL_ON or OFF for purposes of evaluating the power state of supply sets and power domains.

SYNTAX

```
status set_partial_on_translation
  [default_translation]
  [-full_on_tools {tools}]
  [-off_tools {tools}]
```

Data Types

<i>default_translation</i>	string
<i>tools</i>	list

ARGUMENTS

default_translation

Specifies the default translation for unlisted tools. Valid values are FULL_ON, OFF and UNDETERMINED.

-full_on_tools {*tools*}

Specifies the tools in which PARTIAL_ON is translated to FULL_ON. The tool names are determined by simulation tool and implementation tool doesn't need to do any tool name check for the specified value.

-off_tools {*tools*}

Specifies the tools in which PARTIAL_ON is translated to OFF. The tool names are determined by simulation tool and implementation tool doesn't need to do any tool name check for the specified value.

DESCRIPTION

This command defines the translation of PARTIAL_ON to FULL_ON or OFF for purposes of evaluating the power state of supply sets and power domains. This command does not affect implementation and is intended to be used by simulation tools, to read and write transparently.

EXAMPLES

The following example sets the `partial_on_translation` to `FULL_ON` for the *Design-Compiler* tool.

```
prompt> set_partial_on_translation OFF \  
      -full_on_tools {Design-Compiler}
```

```
prompt> add_power_state FULL_ON \  
      -off_tools {OTHER-TOOL}
```

SEE ALSO

`create_power_switch(2)`

set_path_margin

Specifies a margin to adjust required times for specified paths in the current design.

SYNTAX

```
status set_path_margin
    margin_value
    [-rise | -fall]
    [-setup | -hold]
    [-from from_list
     | -rise_from rise_from_list
     | -fall_from fall_from_list]
    [-through through_list
     | -rise_through rise_through_list
     | -fall_through fall_through_list]
    [-to to_list
     | -rise_to rise_to_list
     | -fall_to fall_to_list]
    [-reset_path]
```

Data Types

<i>margin_value</i>	float
<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list
<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list

ARGUMENTS

margin_value

Specifies the value of the margin by which the required time is adjusted for the specified paths. A positive value results in a more restrictive or tighter check. A negative value results in a less restrictive or looser check. The value must be specified in the same time units as those of the technology library used during optimization.

-rise | -fall

Applies the timing adjustment only to paths having a rising signal or only to paths having a falling signal at the path endpoint. By

default, paths having either a rising or falling signal at the path endpoint are adjusted.

-setup | -hold

Applies the timing adjustment to only setup or only hold timing constraints for the specified paths. By default, both setup and hold constraints are adjusted.

-from *from_list*

Specifies a list of path startpoints (port, pin, clock, or cell names) of the current design. If you specify a clock, all path startpoints related to that clock are affected. If you specify a cell name, one path startpoint on that cell is affected. All paths from these startpoints to the endpoints in the *to_list* are adjusted by the *margin_value*. If you don't specify *to_list*, all paths from *from_list* are affected. This list cannot include output ports. If you include more than one object, you must enclose the objects in quotation marks (") or braces ({}).

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from *fall_from_list*

Same as the **-rise_from** option, except that the path must fall from the objects specified.

-through *through_list*

Determines a list of path throughpoints (port, pin, or leaf cell names) of the current design. The margin value applies only to paths that pass through one of the points in the *through_list*. If you include more than one object, you must enclose the objects in quotation marks (") or braces ({}). If you specify the **-through** option multiple times, the margin values apply to paths that pass through a member of each *through_list* in the order in which the lists were given. In other words, the path must first pass through a member of the first *through_list*, then through a member of the second list, and so on, for every through list specified. If you use the **-through** option in combination with the **-from** or **-to** option, the margin adjustment applies only if the **-from** or **-to** conditions are satisfied and the **-through** conditions are satisfied.

-rise_through *rise_through_list*

Same as the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than once in a single command invocation as with the **-through** option.

-fall_through *fall_through_list*

Same as the **-through** option, but applies only to paths with a falling transition at the specified objects. You can specify **-fall_through** more than once in a single command invocation as with the **-through** option.

-to *to_list*

Specifies a list of path endpoints (port, clock, cell, or pin names) of the current design. All paths to the endpoints in the *to_list* are adjusted by *margin_value*. If you don't specify a *from_list*, all paths to *to_list* are affected. This list cannot include input ports. If you include more than one object, you must enclose the objects in quotation marks (") or braces ({}). If you specify a cell, one path endpoint on that cell is affected. If you specify a clock, all path endpoints related to that clock are affected.

-rise_to *rise_to_list*

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path.

-fall_to *fall_to_list*

Same as the **-rise_to** option, but applies only to paths falling at the endpoint.

You can use no more than one of **-to**, **-rise_to**, and **-fall_to**.

-reset_path

Removes existing point-to-point exception information on the specified paths. Only information of the same rise/fall setup/hold type is reset. This is equivalent to using the **reset_path** command with similar options before executing the **set_path_margin** command.

DESCRIPTION

This command adjusts the data required time for specified paths by a specified amount, *margin_value*. The required time for any startpoint in *from_list* to any endpoint in *to_list* is adjusted by *margin_value*. A positive margin value results in a more restrictive or tighter check, whereas a negative margin value results in a less restrictive or looser check. Based on the type of timing constraint, either setup or hold, the margin value is either subtracted from or added to the default data required time.

The **set_path_margin** command is a point-to-point timing exception command; that is, it adjusts the required time of the path and therefore the endpoint slack for the timing paths.

Other point-to-point timing exception commands include the **set_multicycle_path**, **set_min_delay**, **set_false_path**, and **set_max_delay** commands.

If a path satisfies multiple timing exceptions, the following rules are applied to determine which exceptions take effect.

Rules referring to **-from** apply equally to **-rise_from** and **-fall_from**, and similarly for the rise and fall options of **-through** and **-to**.

1. If one exception is **set_false_path**, and the other is **set_path_margin**, then **set_false_paths** takes precedence.
2. If one exception is **set_path_margin** and the other is **set_max_delay**, **set_min_delay**, or **set_multicycle_path**, no conflict occurs and both constraints are honored.
3. If one exception has a **-from pin** or **-from cell** and the other does not, the former takes precedence.
4. If one exception has a **-to pin** or **-to cell** and the other does not, the former takes precedence.
5. If one exception has any **-through** points and the other does not, the former takes precedence.
6. If one exception has a **-from clock** and the other does not, the former takes precedence.
7. If one exception has a **-to clock** and the other does not, the former takes precedence.
8. If you apply **set_endpoint_margin** on an endpoint and also a **set_path_margin** on the same endpoint, then the endpoint required time is adjusted for both the values.

Note: Specifying a **path_margin** for a pin that is not a path endpoint places a **size_only** attribute on that cell.

Use the **report_timing_requirements** command to list the **path_margin**, **max_delay**, **min_delay**, **multicycle_path**, and **false_path** information for the design.

To undo the effects of the **set_path_margin**, use the **reset_path** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example adjusts the required time to a port named Y by 10 units. The adjustment results in a more restrictive, tighter timing constraint because the margin value is positive.

```
prompt> set_path_margin 10.0 -to {Y}
```

The following example specifies that the required time of all paths from cell ff1a or ff1b that pass through cell u1 and end at cell ff2e are to be adjusted by 15.0 units.

```
prompt> set_path_margin 15.0 -from {ff1a ff1b} -through {u1} -to {ff2e}
```

The following example shows how to specify that all paths to endpoints clocked by PHI2 are to be adjusted by 8.5 units.

```
prompt> set_path_margin 8.5 -to [get_clocks PHI2]
```

The following example specifies that all timing paths from pin ff1/CP to pin ff2/D that rise through at least one of pins U1/Z and U2/Z and fall through at least one of pins U3/Z and U4/C are to be adjusted by 8.0 units.

```
prompt> set_path_margin 8.0 -from {ff1/CP} -to {ff2/D} \  
-rise_through {U1/Z U2/Z} -fall_through {U3/Z U4/C}
```

SEE ALSO

- compile(2)
- create_clock(2)
- group_path(2)
- report_constraint(2)
- report_path_group(2)
- reset_design(2)
- reset_path(2)
- set_false_path(2)
- set_input_delay(2)
- set_min_delay(2)
- set_multicycle_path(2)
- set_output_delay(2)

set_pg_pin_model

Defines the power and ground pins for a library cell.

SYNTAX

```
status set_pg_pin_model
  cell_name
  [-pg_pin_name pin_names]
  [-pg_voltage_name voltage_names]
  [-pg_pin_type pin_types]
  [-pg_pin_direction pin_directions]
  [-pg_physical_connection physical_connections]
  [-pg_related_bias_pin related_bias_pins]
```

Data Types

<i>cell_name</i>	string
<i>pin_names</i>	list
<i>voltage_names</i>	list
<i>pin_types</i>	list
<i>pin_directions</i>	list
<i>physical_connections</i>	list
<i>related_bias_pins</i>	list

ARGUMENTS

cell_name

Specifies the name of the library cell for which to define the power and ground pins.

-pg_pin_name *pin_names*

Specifies the names of the power and ground pins of the library cell.

-pg_voltage_name *voltage_names*

Specifies the voltage name of the corresponding power or ground pin of the library cell. The voltage names are defined in the library voltage map.

There must be a one-to-one correspondence between the voltage names specified in this option and the pins specified in the **-pg_pin_name** option.

-pg_pin_type *pin_types*

Specifies the pin type for the each pin specified in the **-pg_pin_name** option. Valid values are `primary_power`, `primary_ground`, `backup_power`, `backup_ground`, `internal_power`, `internal_ground`, `pwell`, `nwell`, `deepnpwell`, and `deepnwell`.

There must be a one-to-one correspondence between the pin types specified in this option and the pins specified in the **-pg_pin_name** option.

-pg_pin_direction *pin_directions*

Specifies the direction for each pin specified in the **-pg_pin_name** option. Valid values are input, output, inout, and internal.

There must be a one-to-one correspondence between the pin directions specified in this option and the pins specified in the **-pg_pin_name** option.

-pg_physical_connection *physical_connections*

Specifies the type of physical connection used for each pin specified in the **-pg_pin_name** option. Valid values are device_layer and routing_pin.

There must be a one-to-one correspondence between the connection types specified in this option and the pins specified in the **-pg_pin_name** option.

-pg_related_bias_pin *related_bias_pins*

Specifies the related bias pin for each pin specified in the **-pg_pin_name** option.

There must be a one-to-one correspondence between the related bias pins specified in this option and the pins specified in the **-pg_pin_name** option.

DESCRIPTION

The **set_pg_pin_model** command defines the power and ground pins for a library cell. If the power or ground pin already exists, the new specification overrides the old one. Otherwise, new power and ground pins are created for the library cell.

EXAMPLES

The following example defines the power and ground pins for the library cells whose name starts with LVLH.

```
prompt> set_pg_pin_model LVLH* -pg_pin_name {VDD VSS} \  
-pg_voltage_name {VDDL VSS} \  
-pg_pin_type {primary_power primary_ground}
```

SEE ALSO

set_voltage_model(2)
report_lib(2)
check_library(2)

set_physical_hierarchy

Sets a list of cells to be physical hierarchies. This command is supported only in Design Compiler topographical mode.

SYNTAX

```
status set_physical_hierarchy  
      object_list
```

Data Types

object_list list

ARGUMENTS

object_list

Specifies the list of hierarchical cells to be set as a physical hierarchy.

DESCRIPTION

The **set_physical_hierarchy** command sets cells in the object list as physical hierarchies. By default, a physical hierarchy has the **dont_touch** attribute.

EXAMPLES

The following example sets the HIER_1 hierarchical cell to be a physical hierarchy:

```
prompt> set_physical_hierarchy HIER_1
```

SEE ALSO

`get_physical_hierarchy(2)`
`set_cell_location(2)`

set_pin_model

Specifies the related power and ground pins for a library cell.

SYNTAX

```
status set_pin_model
      cell_name
      [-pins pin_names]
      [-related_power_pin pin_names]
      [-related_ground_pin pin_names]
      [-related_bias_pin pin_names]
      [-power_down_function functions]
```

Data Types

```
cell_name  string
pin_names  list
functions  list
```

ARGUMENTS

cell_name

Specifies the name of the library cell for which to define the related power and ground pins.

-pins *pin_names*

Specifies the names of the pins for which you are defining the related power and ground pins.

-related_power_pin *pin_names*

Specifies the related power pin for each pin specified in the **-pins** option.

There must be a one-to-one correspondence between the power pins specified in this option and the pins specified in the **-pins** option.

-related_ground_pin *pin_names*

Specifies the related ground pin for each pin specified in the **-pins** option.

There must be a one-to-one correspondence between the ground pins specified in this option and the pins specified in the **-pins** option.

-related_bias_pin *pin_names*

Specifies the related bias pin for each pin specified in the **-pins** option.

There must be a one-to-one correspondence between the bias pins specified in this option and the pins specified in the **-pins** option.

-power_down_function *functions*

Specifies the power-down function for each pin specified in the **-pins** option.

There must be a one-to-one correspondence between the function specified in this option and the pins specified in the **-pins** option.

DESCRIPTION

The **set_pin_model** command defines the related power pin, related ground pin, and related bias pin for the specified library cell pins. If the related pin is already defined, the new specification overrides the old one.

EXAMPLES

The following example defines the pin model for the RETN and RETNOUT pins of the library cells that match the pattern *DRFF*.

```
prompt> set_pin_model *DRFF* -pins {RETN RETNOUT} \  
-related_power_pin VDDG \  
-related_ground_pin VSSG
```

The following example defines the pin model for the SLEEPOUT pin of the library cells that match the pattern HEAD*.

```
prompt> set_pin_model HEAD* -pins {SLEEPOUT} \  
-power_down_function "!VDDG + VSS"
```

SEE ALSO

set_voltage_model(2)
set_pg_pin_model(2)
report_lib(2)
check_library(2)

set_pin_name_synonym

Defines synonyms for pin names.

SYNTAX

```
status set_pin_name_synonym
  [-full_name]
  [-force]
  pin_name_synonym
  pin_name
```

Data Types

```
pin_name_synonym  string
pin_name          string
```

ARGUMENTS

-full_name

Indicates that a pin name synonym is a full name synonym.

-force

Overwrites the existing pin name synonym if a conflict occurs.

pin_name_synonym

Specifies a pin name synonym string for *pin_name*. This argument is required.

pin_name

Specifies a pin name string for which the synonym is defined. This argument is required.

DESCRIPTION

This command defines pin name synonyms, which are supported in pin object queries.

The *pin_name_synonym* is an alternative pin name for a pin object. For example, if the U1/U2/data_in name is defined as the synonym for the U1/U2/D pin, pin query commands can find the pin object named U1/U2/D when the commands could not find the pin with the given name U1/U2/data_in.

Pin name synonyms are only applicable to pins on leaf-level sequential cells. This feature can also be applicable to physical and logic library pin names.

The tool supports two types of pin name synonyms: full name synonyms and simple name synonyms. The **set_pin_name_synonym** command defines simple name synonyms unless the **-full_name** option is used. When defining a full name synonym, you specify the full hierarchical name for the pin object in both the *pin_name_synonym* and *pin_name* fields. Pin query commands use full name synonyms exactly as they are defined. Simple name synonym definitions use short names for pin objects. For library pin, only the simple name is needed. Pin query commands use a simple pin name synonym to construct a full pin name during a synonym-based pin object search.

You can get full name and simple name synonyms for pin objects by querying the **full_name** and **name** attributes for pin objects.

Wildcards and regular expressions are not supported in pin name synonym definitions. Simple name synonyms do not support the forward slash (/) since it is considered a hierarchical separator.

When you use the pin name synonyms for pin queries, the full name synonym, when applicable, supersedes the simple name synonym.

EXAMPLES

The following examples use **set_pin_name_synonym** to define pin name synonyms and show how **get_pins** command uses the synonyms:

```
prompt> set_pin_name_synonym SYN_CD CD
1
```

```
prompt> set_pin_name_synonym -full_name \
this/is/a/synonym mid1/FJK2SP_at_mid/T1
1
```

```
prompt> set_pin_name_synonym -full_name \
mid1/FJK2SP_at_mid/J mid2/bot2/LSR0P_at_bot/Q
1
```

SEE ALSO

remove_pin_name_synonym(2)
report_pin_name_synonym(2)

set_pin_physical_constraints

Sets physical constraints for pin instances.

SYNTAX

```
status set_pin_physical_constraints
  objects
  | -pin_name pin_name [-cell cell_name]
  | -nets nets [-cell cell_name]
  [-layers layers]
  [-width pin_width]
  [-depth pin_depth]
  [-side side_number]
  [-offset offset_distance]
  [-order order_number]
  [-pin_spacing spacing]
  [-exclude_sides exclude_side_numbers]
  [-off_edge center | location | auto]
  [-location point]
```

Data Types

<i>objects</i>	collection
<i>pin_name</i>	string
<i>cell_name</i>	string
<i>nets</i>	collection
<i>layers</i>	list
<i>pin_width</i>	float
<i>pin_depth</i>	float
<i>side_number</i>	integer
<i>offset_distance</i>	float
<i>order_number</i>	integer
<i>spacing</i>	integer
<i>exclude_side_numbers</i>	list of integers

ARGUMENTS

objects

Specifies the pins to which the specified constraints are applied. In most cases, you can specify multiple objects. However, if you specify the **-order** option, you can specify only a single pin.

The *objects* argument, **-pin_name** option, and **-nets** option are mutually exclusive. You must specify the pins by using only one of these arguments.

In Design Compiler Topographical mode, only port objects are supported.

-pin_name *pin_name*

In Design Compiler Topographical mode, the command issues an error to indicate that the option is not supported.

-cell *cell_name*

In Design Compiler Topographical mode, the command issues an error to indicate that the option is not supported.

-nets *nets*

In Design Compiler Topographical mode, the command issues an error to indicate that the option is not supported.

-layers *layers*

In Design Compiler Topographical mode, if you specify one layer, the command places the pin on that layer. If you specify more than one layer, the command issues an error to indicate that multiple layers are not supported.

-width *pin_width*

Specifies the pin width in microns. If the specified width constraint is smaller than the minimum width given in the library, the minimum width is applied (unless otherwise noted). The default is 0.

-depth *pin_depth*

Specifies the depth in microns. If the depth constraint specified is smaller than minLength defined in the tech file, then the value of minLength will be honored. If minArea is also defined in the tech file, then the value of pin depth times pin width should also honor minArea. The default is 0.

-side *side_number*

Specifies the block side on which the pin must be placed so that it abuts the specified die edge.

The side number is a positive integer that starts from 1. Given any shape (rectangular or rectilinear), the lower left-most vertical edge is the starting edge (side number 1). The side number of the next edge, going clockwise, is 2, and so on. You can specify a value of 0 to indicate no side constraint.

-offset *offset_distance*

Specifies the distance in microns that the pins must be offset from the starting point of the specific edge. The offset is the distance to the edge of the pin. For a horizontal edge, the starting point is the left vertex of the edge. For a vertical edge, the starting point is the lower vertex of the edge. You must specify a positive floating point value for the offset distance.

-order *order_number*

Specifies the relative placement order for the specified pin. The order number must be a positive integer. A smaller order number specifies a more left side placement on a horizontal placement edge, and a more bottom side placement on a vertical placement edge. An order number of 0 specifies that the pin has no order constraint. Order-constrained pins can be interleaved by other pins. Constraints specified by the **-order** option do not apply to feedthrough pins. The default order number is 0.

-pin_spacing *spacing*

Specifies the minimum number of wire tracks between adjacent pins or between a pin and a preroute wire that cuts across a soft macro or plan group edge in the normal direction. Note that pins that are created are always snapped to wire tracks. The default minimum pin-to-pin spacing is one wire track.

-exclude_sides *exclude_side_numbers*

Specifies the soft macro edges on which pins cannot be placed. The argument is a string consisting of a number or a set of numbers separated by commas. Each number corresponds to a macro or plan group edge. The lower leftmost vertical edge is numbered 1 (one). Other edges are numbered according to their order of traversal in the clockwise direction starting from edge 1.

If there are multiple leftmost edges then the edge 1 is the lowest leftmost edge. For example, to exclude sides 2, 4, and 6, enter

```
-exclude_sides {2,4,6}
```

-off_edge center | location | auto

Specifies that the pin is placed off-edge. By default, pins are placed on-edge. The **center**, **location** or **auto** keyword specifies how the pin is placed. You must specify only one keyword.

- **center**
When you specify **-off_edge center**, the pin is placed at the center of the cell.
- **location**
In Design Compiler Topographical mode, the command issues an error to indicate that the location value is not supported.
- **auto**
When you specify **-off_edge auto**, you must create a pin guide that is associated with this pin. The pin guide must be in the center of the block boundary. The pin is placed at the optimal position guided by the pin guide. If you do not provide a pin guide, the pins are placed in the center of the cell, the result is the same as if you specified **-off_edge center**.

-location point

In Design Compiler Topographical mode, the command issues an error to indicate that the option is not supported.

DESCRIPTION

This command sets physical constraints, such as layer and width, on the specified pins. The constraints you specify are saved when the design is saved.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies physical constraints on a pin.

```
prompt> set_pin_physical_constraints [get_port "Reset"] \
  -layers {M2} \
  -width 0.28 -depth 0.28 -side 1 -offset 5
```

SEE ALSO

set_pipeline_scan_data_configuration

Specifies the pipelined scan data configuration for the design.

SYNTAX

```
int set_pipeline_scan_data_configuration
  [-head_pipeline_clock clock_name]
  [-tail_pipeline_clock clock_name]
  [-head_pipeline_stages total_depth]
  [-tail_pipeline_stages total_depth]
  [-head_shared_pipeline_stages shared_top_level_depth]
  [-tail_shared_pipeline_stages shared_top_level_depth]
  [-head_scan_flop true | false]
```

Data Types

<i>clock_name</i>	string
<i>total_depth</i>	integer
<i>shared_top_level_depth</i>	integer

ARGUMENTS

-head_pipeline_clock *clock_name*

Specifies the shift clock for the head pipeline scan data registers. The specified clock must be previously declared as a scan clock using the **set_dft_signal** command. All head pipeline registers are triggered by the trailing edge of the clock. If no clock is specified, the tool creates and uses a new clock signal.

If you enable beginning-of-chain leading-edge retiming registers using the **set_scan_configuration -add_test_retiming_flops** command with the **begin_only** or **begin_and_end** option values, the head pipeline registers are clocked on the leading clock edge instead.

In an OCC controller flow (DFT-inserted or user-defined), you can specify the name of an ATE clock that is predefined with the **set_dft_signal** command. This specification is mapped to the first-defined OCC-controlled clock associated with that ATE clock during DFT insertion.

-tail_pipeline_clock *clock_name*

Specifies the shift clock for the tail pipeline scan data registers. The specified clock must be previously declared as a scan clock using the **set_dft_signal** command. All tail pipeline registers are triggered by the leading edge of the clock. If no clock is specified, the tool creates and uses a new clock signal.

OCC clock specification is the same as with the **-head_pipeline_clock** option.

-head_pipeline_stages *total_depth*

Specifies the number of head pipeline stages. The default is 1. You can choose not to insert head pipeline stages by specifying a depth of 0.

-tail_pipeline_stages *total_depth*

Specifies the number of tail pipeline stages. The default is 1. You can choose not to insert tail pipeline stages by specifying a depth of 0.

-head_shared_pipeline_stages *shared_top_level_depth*

Specifies the number of top-level shared head pipeline stages to use when shared codec input connections are used for core integration.

When set, the tool uses the specified number of shared pipeline registers along the shared scan input path, then it uses dedicated pipeline registers as needed for the remaining stages to meet the total head pipeline depth target. A value of 0 uses only dedicated top-level stages. The default is to use the largest number of shared top-level stages possible given the constraints, which minimizes the number of dedicated top-level stages.

This option is only used when shared codec input connections are specified with the **-shared_inputs** option of the **set_scan_compression_configuration** command.

-tail_shared_pipeline_stages *shared_top_level_depth*

Specifies the number of top-level shared tail pipeline stages to use when shared codec output connections are used for core integration.

When set, the tool uses the specified number of shared pipeline registers along the shared scan output path, then it uses dedicated pipeline registers as needed for the remaining stages to meet the total tail pipeline depth target. A value of 0 uses only dedicated top-level stages. The default is to use the largest number of shared top-level stages possible given the constraints, which minimizes the number of dedicated top-level stages.

This option is used only when shared codec input connections are specified with the **-shared_inputs** and **-shared_outputs** options of the **set_scan_compression_configuration** command.

-head_scan_flop *true | false*

When set to **true**, scan-replaced registers are used for the head pipeline stages. The scan data input of each head pipeline scan register is used for the pipeline shift path. The functional data input of each register is driven by that register's output, so that the register holds state during scan capture. The default is **false**, which inserts nonscan head pipeline registers.

SEE ALSO

reset_pipeline_scan_data_configuration(2)
set_dft_configuration(2)
set_scan_compression_configuration(2)

set_placement_area

Creates the core placement area. This command is supported only in topographical mode.

SYNTAX

```
int set_placement_area  
  -coordinate {X1 Y1 X2 Y2}  
  [-fixed | -unfixed]
```

ARGUMENTS

-coordinate {X1 Y1 X2 Y2}

{X1 Y1 X2 Y2} specify the lower left and upper right coordinates of the core area. The coordinates are in microns relative to the chip origin. This options will imply -fixed flag unless -unfixed is used explicitly.

-fixed

Marks the core area to be fixed. The tool should not change the core area.

-unfixed

Reset the fixed flag.

DESCRIPTION

The **set_placement_area** command defines the core placement area for rough placement. This will overwrite the previous core area.

This command can also be used to mark or unmark "fixed_core" attribute which indicate the core area may or may not be changed by the tool.

EXAMPLES

The following example shows how to set the placement area to the rectangle with lower left corner (100,100) and upper right corner (2000,2000).

```
prompt> set_placement_area -coordinate { 100 100 2000 2000}
```

SEE ALSO

set_port_attributes

Sets the specified attributes and their value on the ports.

SYNTAX

```
status set_port_attributes
  [-ports port_list]
  [-elements element_list]
  [-exclude_ports port_list]
  [-exclude_elements element_list]
  [-applies_to inputs | outputs | both]
  [-attribute at_name at_value]
  [-receiver_supply supply_set_ref]
  [-driver_supply supply_set_ref]
  [-repeater_supply supply_set_ref]
  [-model model_name]
  [-feedthrough]
  [-unconnected]
  [-clamp_value clamp_value]
  [-is_analog]
```

Data Types

<i>port_list</i>	list
<i>element_list</i>	list
<i>at_name</i>	string
<i>at_value</i>	string
<i>supply_set_ref</i>	string
<i>model_name</i>	string
<i>clamp_value</i>	string

ARGUMENTS

-ports *port_list*

Specifies the collection of ports on which the attributes must be set. Wildcards are accepted in port names.

-elements *element_list*

Specifies the current scope or list of instance names for whose ports or pins the attributes are applicable. Wildcards are accepted in the instance names.

One of the **-ports** or **-elements** options must be specified for this command.

-exclude_ports *port_list*

Specifies the collection of ports on which the attributes should not be applied.

-exclude_elements *element_list*

Specifies the current scope or list of instance names for whose ports or pins the attributes should not be applied.

-applies_to inputs | outputs | both

Specifies whether the given **set_port_attributes** command applies to all input ports or output ports or both input and output ports of the specified element. The argument must be used in conjunction with the **-elements** argument. The default for this argument is **both**.

-attribute *at_name at_value*

Specifies the name and value of the attribute to be set on the ports specified by the **-ports** option.

The supported attribute names are as follows:

- **iso_source** (used on elements and signal ports)
- **iso_sink** (used on elements and signal ports)
- **snps_derived** (used on supply ports only)
- **related_supply_default_primary** (used on elements, implies to their ports or pins)
- **repeater_power_net** (used on domain root cells and domain boundary ports/pins)
- **repeater_ground_net** (used on domain root cells and domain boundary ports/pins)

The supported values are as follows:

- For the **iso_source** and **iso_sink** attributes, the value must be the name of a supply set that is already defined.
- For the **snps_derived** attribute, the value must be **true** or **false**, where **true** implies that the attributed supply port has been derived internally by the tool.
- For the **related_supply_default_primary** attribute, only a Boolean value of **true** is supported.
- For **repeater_power_net** attribute, the value must be a supply net.
- For **repeater_ground_net** attribute, the value must be a supply net.

-receiver_supply *supply_set_ref*

Specifies the supply for the logic reading the port.

-driver_supply *supply_set_ref*

Specifies the supply for the logic driving the port.

-repeater_supply *supply_set_ref*

Specifies the supply used by the repeater driving the port.

-model *model_name*

Specifies a module or library cell on whose ports the attribute is going to be applied.

-feedthrough

Indicates that the specified ports are connected together internally to form a feedthrough path. The **-feedthrough** option must be used in conjunction with the **-model** and **-ports** options, and is mutually exclusive to the **-unconnected** option.

-unconnected

Indicates that the specified ports are not connected internally. The **-unconnected** option must be used in conjunction with the **-model** and **-ports** options, and is mutually exclusive to the **-feedthrough** option.

-clamp_value *clamp_value*

Specifies the clamp value to be used on ports that have an isolation strategy.

DESCRIPTION

This command sets the attributes and their value on a list of ports specified by the **-ports** option or on a list of instances specified by the **-elements** option.

Specifying more than one **-attribute** option is supported only for repeater supply net attributes.

Except for repeater supply net attributes, all other attributes can be set multiple times on the same ports. The last value specified is considered.

If an attribute is defined with both the **-ports** and **-elements** options, the one specified with the **-ports** option has precedence over the one specified with the **-elements** option regardless of the order in which they are specified.

The **iso_source** attribute can only be specified on an input port of the design. It specifies the outside source supply to be used to determine if an isolation strategy with the source or **diff_supply_only** options should insert an isolation cell.

The **iso_sink** attribute can only be specified on an output port of the design. It specifies the outside sink supply to be used to determine if an isolation strategy with the sink or **diff_supply_only** options should insert an isolation cell.

The **related_supply_default_primary** attribute can only be used with the **-elements** option and can be set on cell instances. For ports without the **set_related_supply_net** setting, the tool assumes the primary supply of the ports' domain is the default supply. This assumption can cause equivalence failure when the netlist and the UPF' file is verified with RTL and UPF. Setting this attribute with **-elements** as **{.}** helps avoid the equivalence failure. The tool passes this setting to the UPF' file, thus making sure the equivalence tool also sees the same assumption as synthesis tool. If this attribute is not set, the **compile_ultra** command warns the possible synthesis and equivalence check mismatch, on finding any ports without the **set_related_supply_net** setting.

The **repeater_power_net** and **repeater_ground_net** attributes must be specified together in a single **set_port_attributes** command. These attributes cannot be specified on the input ports of the top domain and output pins of the macro domain. It is an error to apply these attributes on ports/pins of direction inout. These attributes are not allowed to be specified with the **-repeater_supply** option.

The **-receiver_supply** and **-driver_supply** options can be used on the top-level ports, the design, or black box cells and their pins. A black box cell must be the root cell of a power domain. In all other cases, the command reports and ignores such objects. The supply of the visible logic specified by these options is preserved.

The **-receiver_supply** and **-driver_supply** options have higher precedence than the **related_supply_net** attribute. You cannot use the **set_related_supply_net** command on ports that have the **-receiver_supply** or **-driver_supply** setting.

The **-driver_supply** option is applied to the input ports only if the supply voltage (if already specified for the supply using the **set_voltage** command) matches the voltage of the driving cell (as specified by the **set_driving_cell** command for the input port). It is an error if the voltages do not match, and the command ignores the ports that do not meet the voltage compliance check.

The **-repeater_supply** option can be specified on any domain root cells or boundary pins except input ports of the top domain and output pins of the macro domain. It is an error to use the **-repeater_supply** option on ports/pins of direction inout. This option is not allowed to be specified with **repeater_power_net** and **repeater_ground_net** attributes.

The **-clamp_value** option can be specified on any **-ports** or **-elements** where an isolation strategy can be applied. It specifies the

clamp value that should be used when isolating the referenced ports/elements. Supported values are **0**, **1** and **latch**. In case of multiples instances of the same command, but with a different **-clamp_value** value, the last setting is preserved.

The **-literal_supply** option can be applied to the input pins of macro cells, top-level output ports and hierarchical cells and their pins. In case of multiples instances of the command in the same port, but with a different **-literal_supply** value, the highest precedence setting is preserved. If settings with the same precedence are applied, the last one is preserved.

The **-model** option can only be used with the **-feedthrough** or **-unconnected** options. Otherwise, it is an error.

When using the **-feedthrough** or **-unconnected** options, the model referenced with the **-model** option must be either a library cell that is a PAD or MACRO, or a module that is a black box.

One of the **-attribute**, **-driver_supply**, **-receiver_supply**, **-repeater_supply**, **-feedthrough**, **-unconnected**, **-clamp_value** or **-literal_supply** options is required for this command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the **snps_derived** attribute to **true** on the **power_port** and **ground_port** ports in the block named **instA**.

```
prompt> set_port_attributes \
  -ports {instA/power_port instA/ground_port} \
  -attribute snps_derived true
1
```

The following examples set the **iso_source** and **iso_sink** attributes on the input and output ports respectively.

```
prompt> set_port_attributes -ports {in1 in2} -attribute iso_source SS1
prompt> set_port_attributes -ports {out1 out2} -attribute iso_sink SS2
```

The following example sets the **related_supply_default_primary** attribute on the design element corresponding to the current scope represented by the dot.

```
prompt> set_port_attributes -elements {.} \
  -attribute related_supply_default_primary true
Information: Resolving '.' to the current scope 'top'.
```

The following example sets the **related_supply_default_primary** attribute on the cell instances **U1** and **U2/U3**.

```
prompt> set_port_attributes -elements {U1 U2/U3} \
  -attribute related_supply_default_primary true
```

The following example sets the **repeater_power_net** and the **repeater_ground_net** attributes on the cell instances **U1** which must be a power domain root cell.

```
prompt> set_port_attributes -elements {U1} \
  -attribute repeater_power_net VDD -attribute repeater_ground_net VSS
```

The following example sets the **repeater_supply** attribute on the boundaries of the top power domain.

```
prompt> set_port_attributes -elements {.} \
  -repeater_supply SS1 -applies_to both
Information: Resolving '.' to the current scope 'top'.
```

The following example shows the error message when multiple **-attribute** options are used with attributes other than repeater supply net attributes.

```
prompt> set_port_attributes -ports {in1 out1} \
  -attribute iso_source SS1 -attribute iso_sink SS2
Error: Attribute format is not acceptable: Multiple -attribute are allowed only
for repeater supply net attributes. (UPF-210)
```

```
prompt> set_port_attributes -ports {in1 out1} \
  -attribute repeater_power_net VDD -attribute iso_sink SS2
Error: Attribute format is not acceptable: Dissimilar attributes specified. (UPF-210)
```

The following example sets the receiver and driver supply attributes for the ports of the design element corresponding to the current scope. The current scope in the following examples is that of the design 'top'.

```
prompt> set_port_attributes -elements { . } \
  -receiver_supply SS1 -applies_to outputs
Information: Resolving '.' to the current scope 'top'.
```

```
prompt> set_port_attributes -elements { . } \
  -driver_supply SS1 -applies_to inputs
Information: Resolving '.' to the current scope 'top'.
```

The following example issues UPF-231 warning message when you set the driver supply attribute on a hierarchical cell u1.

```
prompt> set_scope u1
prompt> set_port_attributes -elements { . } \
  -receiver_supply SS1 -applies_to outputs
Information: Resolving '.' to the current scope 'u1'.
Warning: Invalid object(s) specified for '-receiver_supply/-driver_supply'
attribute.
Reason: Only the top design, or black box is allowed.
Ignoring 1 invalid object(s) {u1}. (UPF-231)
```

The following example creates a feedthrough path between two pins of a library cell

```
prompt> set_port_attributes -model {macro_lib_cell} -ports {in out} -feedthrough
```

The following example marks a pin as internally unconnected on a library cell

```
prompt> set_port_attributes -model {macro_lib_cell} -ports {unused} -unconnected
```

The following example sets the clamp value on a hierarchical instance's input pins and then changes the setting on a specific pin.

```
prompt> set_port_attributes -elements {hierarchy_name} -clamp_value 0 -applies_to inputs
prompt> set_port_attributes -ports {hierarchy_name/input_port} -clamp_value 1
```

The following example sets the clamp value on a hierarchical pin and the overrides it with a different clamp value.

```
prompt> set_port_attributes -ports {hierarchy_name/input_port} -clamp_value 1
prompt> set_port_attributes -ports {hierarchy_name/input_port} -clamp_value 0

prompt> set_port_attributes -ports {macro_cell/in} -literal_supply MACRO_SS1
prompt> set_port_attributes -elements { . } -literal_supply {VDDtop VSSStop} -applies_to outputs
```

SEE ALSO

`insert_mv_cells(2)`
`set_related_supply_net(2)`
`set_isolation(2)`

set_port_fanout_number

Sets the number of external fanout points driven by specified ports in the current design.

SYNTAX

```
status set_port_fanout_number  
      fanout_number  
      port_list
```

Data Types

```
fanout_number  float  
port_list      list
```

ARGUMENTS

fanout_number

Specifies the number of external fanout points driven by the ports in *port_list*. Use this command only with ports. An error message occurs if *port_list* contains any nets. The input value is interpreted as an integer, non-integer values are truncated, and the integer portion is used as the fanout number.

port_list

Specifies a list of ports in the current design whose fanout numbers are to be set.

DESCRIPTION

This command sets the **fanout** attribute on specified ports in the current design. The fanout attribute sets the number of external fanout points driven by each port. This information is used to calculate an external wire load value for the corresponding port. The external wire load value is then added to the wire load of the net connected to the port. Setting this value to 0 when the driver of the port has no other internal fanout causes the net driving the port to be treated as unconnected, so it may be skipped for DRC.

Use the **report_port** command to view the load values on ports.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following sequence of commands describes the external fanout of an output port:

```
prompt> set_port_fanout_number 5 [get_ports O1]
```

```
prompt> set_wire_load_model [get_ports O1] \  
"default_wl"
```

SEE ALSO

- all_outputs(2)
- current_design(2)
- link(2)
- load_of(2)
- remove_attribute(2)
- report_port(2)
- reset_design(2)
- set_drive(2)
- set_load(2)
- target_library(3)

set_port_location

Annotates the specified top-level port with x- and y-coordinates and layer geometry, which the tool uses when running the **reoptimize_design** command.

SYNTAX

```
status set_port_location
  [-coordinate {x y}]
  [-layer_name layer_name]
  [-layer_area {lx ly ux uy}]
  [-append]
  port_name
```

Data Types

<i>x</i>	float
<i>y</i>	float
<i>layer_name</i>	string
<i>lx</i>	float
<i>ly</i>	float
<i>ux</i>	float
<i>uy</i>	float
<i>port_name</i>	string

ARGUMENTS

-coordinate {x y}

Specifies the absolute location, in microns, of the x- and y-coordinates of the port.

-layer_name *layer_name*

Specifies the layer name of the top-level design port whose layer geometry is to be annotated.

-layer_area {lx ly ux uy}

Specifies the layer geometry of the port, in microns.

-append

Appends the location of port shapes to the same port. If this option is omitted, the location provided by the command overwrites the existing location.

port_name

Specifies the name of the top-level design port whose location is to be annotated.

DESCRIPTION

This command annotates the port location on the specified port. Executing this command without coordinates resets the port location. You can also set the port layer geometry by specifying layer name and layer area coordinates.

The location of the port specified with this command takes precedence over any annotation from a PDEF file using the **read_pdef** or **read_clusters** command. However, if the port location is reset by executing the **set_port_location** command without the x- or y-coordinates, any annotation from the PDEF file is used during optimization.

Layer area geometry is relative to the port origin.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example annotates port Z, a top-level port in the current design, with the location at 100, 5000.

```
prompt> set_port_location -coord {100 5000} Z
```

The following example resets the location of port Z. The tool uses physical information from any PDEF file to determine the location of port Z.

```
prompt> set_port_location Z
```

The following example sets the layer geometry of rectangle 10*20 of port Z.

```
prompt> set_port_location -layer_name METAL1 \  
-layer_area {-5 -10 5 10} Z
```

SEE ALSO

set_port_side

Specifies the port side constraints for core generation. This command is supported only in topographical mode.

SYNTAX

```
int set_port_side  
    port_list  
    -side {l | r | b | t | number}
```

Data Types

port_list list

ARGUMENTS

port_list

Specifies the list of ports that this command applies to.

-side {l | r | b | t | number}

Specifies the side to which the ports should be snapped. For a rectangular core outline, specify one of the following values: **l** (left), **t** (top), **b** (bottom), and **r** (right). For a rectilinear core outline, specify the edge number as specified by points on the rectilinear outline.

By default, there is no side constraint on the ports, and they are snapped to the nearest side of the block.

DESCRIPTION

The command **set_port_side** specifies the port side constraints for ports. The constraints will be used to generate a coarse floorplan during synthesis.

If the port side constraints are provided, the ports will be snapped to the specified side. Otherwise, in default, the ports will be snapped to the side nearest to the port location assigned by the coarse placer.

Note that if ports with side constraints also have locations constraints set by **set_port_location**, the location constraints have higher priority during synthesis and side information will be ignored. In this case, the side information will be neither reported by **report_physical_constraints** nor written out by **write_physical_constraints**.

Use this command after loading the physical library and design, but before running synthesis commands.

If the core outline is provided then you can use the edge number as constraints in this command. Please see example below.

EXAMPLES

The following example shows how to constrain the ports `clk` and `reset` to be snapped to the left side of the block.

```
prompt> set_port_side {clk reset} -side left
```

The following example shows how to constraint the port on the right lower edge of the L - shaped rectilinear outline

```
prompt> set_rectilinear_outline -coordinate { 0 0 100 0 100 50 50 50 50 100 0 100 }
```

```
prompt> set_port_side {ABC} -side 2
```

SEE ALSO

- `set_port_location(2)`
- `report_physical_constraints(2)`
- `write_physical_constraints(2)`

set_power_clock_scaling

Specify clock scaling for power analysis.

SYNTAX

```
int set_power_clock_scaling  
    [-period period_value]  
    [-ratio ratio_value]  
    [clock_objects]
```

Data Types

<i>period_value</i>	float
<i>ratio_value</i>	float

ARGUMENTS

-period *period_value*

Specify the period of clock which is used in simulation.

-ratio *ratio_value*

Specify the clock ratio (period_used_in_SAIF / period_used_in_SDC).

clock_objects

Specify a list of clocks in the design.

DESCRIPTION

Clock frequency used in Switching Activity Interchange Format (SAIF) files, from logic simulation, for functional verification can differ from the clock frequency used in Synopsys Design Constraints (SDC) file for timing and power analysis. This command tells Design Compiler what frequency is used in logic simulation so that Design Compiler can scale dynamic power numbers properly.

To enable this feature, set the **power_enable_clock_scaling** variable to *true*.

The *clock_objects* option in this command is a list of clocks in the design that have the specified period or ratio values used in simulation for SAIF generation. Here the ratio is defined as period-in-simulation / period-in-analysis. The command can be used multiple times, once for each clock. If the same clock is used more than once, a warning message is issued and the clock specified last takes effect. The *-period* and *-ratio* options cannot be used simultaneously. If no clock object is specified, only the ratio option is allowed and the ratio value is applied to all the nets/pins/ports that have related clock whose scaling factor undefined or do not have

a related clock.

The tool only scales the switching activities from the SAIF file and activities that are propagated or implied. The command returns 1 on success and 0 on failure.

Note that the scaling result is accumulated since the scaled switching activities in design are saved. For example, if a net has an original toggle rate 1.0 and the *-ratio* option of the **set_power_clock_scaling** command is set to 0.1, the 1st scaling results in the net's toggle rate to 0.1 (the original toggle rate 1.0 is replaced), the second scaling results in 0.01, and so on. To come back to the original toggle rates, you have to use the **reset_switching_activity** command and read the SAIF file again.

Abstract blocks are ignored for clock frequency scaling.

The clock scaling is done only for averaged power.

EXAMPLES

A sample script is listed below. In this case, duration logic simulation the SAIF file mac.saif is generated with clock period 24, so you use **set_power_clock_scaling -period 24 [get_clock clk]**, and you specify SDC clock in mac.sdc (it could be period 12, for example). **report_power** will report the power consumption when the design is operated at SDC clock period 12, rather than logic simulation clock period 24.

```
read_sdc ../src/hdl/gate/mac.sdc
read_parasitics -format spef ../src/annotate/mac.spef
read_saif -input "../sim/mac.saif" -instance "tb/macinst"
```

```
set_power_enable_clock_scaling TRUE
set_power_clock_scaling -period 24 [get_clock clk]
report_power
```

The following sample script shows the usage of *-period* and *-ratio* options. In the below case, all the nets/pins/ports related to the clock specified are scaled to the value defined with *-period* option. The scaling ratio of 2 is only applied to nets/pins/ports that have related clock whose scaling factor undefined or do not have an related clock.

```
set_power_enable_clock_scaling TRUE
set_power_clock_scaling -period 24 [get_clock clk]
report_power
set_power_clock_scaling -ratio 2
report_power
```

SEE ALSO

power_enable_clock_scaling(3)

set_power_guide

Sets an existing exclusive move bound as a power guide or power well. This power guide is used as an always-on power guide.

SYNTAX

```
status set_power_guide
  -name exclusive_movebound_name
  [-guard_band_x horizontal_guard_band_width]
  [-guard_band_y vertical_guard_band_width]
```

Data Types

<i>exclusive_movebound_name</i>	string
<i>horizontal_guard_band_width</i>	integer
<i>vertical_guard_band_width</i>	integer

ARGUMENTS

-name *exclusive_movebound_name*

Specifies the name of an existing exclusive move bound that needs to be set as a power guide or power well.

-guard_band_x *horizontal_guard_band_width*

Facilitates obtaining an integral width for the guard band in the horizontal direction.

-guard_band_y *vertical_guard_band_width*

Facilitates obtaining an integral width for the guard band in the vertical direction.

DESCRIPTION

This command sets an existing exclusive move bound as a power guide. Currently this command is used only for setting an exclusive move bound as an always-on power guide or power well.

The always-on power guides in shutdown voltage areas are used for placing regular cells on the always-on paths so that those cells become always-on.

If no voltage areas exist in the design, you cannot set an exclusive move bound set as a power guide. In a design with voltage area, if an exclusive move bound is in the default voltage area and is set as a power guide, addition or removal of a voltage area might result in incorrect inclusion or exclusion of power guides. You should set power guides only after the design is fixed with respect to the voltage areas in it.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows a flow using the **set_power_guide** command. In this flow, the AO_WELL exclusive move bound physically belongs to the VA_SHUTDOWN voltage area and is set as an always-on power guide.

```
prompt> create_bounds -name AO_WELL -coordinate {10 10 20 20} \  
-exclusive [get_cells A_INST]  
prompt> create_voltage_area -name VA_SHUTDOWN -coordinate {5 5 30 30}  
prompt> set_power_guide -name AO_WELL
```

SEE ALSO

- create_bounds(2)
- create_voltage_area(2)
- unset_power_guide(2)

set_power_prediction

Sets the power prediction mode for **compile_ultra** or **compile_ultra -incremental**. This command is supported only in topographical mode.

SYNTAX

```
int set_power_prediction  
    [true | false]  
    [-ct_references lib_cell_list]
```

ARGUMENTS

true | false

When **true**, sets the power prediction mode. When **false**, turns off the power prediction mode. When no argument is given, **true** is assumed.

-ct_references lib_cell_list

Restricts the list of library cells that can be used for clock tree estimation.

DESCRIPTION

In power prediction mode, **compile_ultra** or **compile_ultra -incremental** tries to correlate the post-synthesis power numbers to that at the end of place and route. Along with improved correlation for power consumption of design components, it predicts the power consumption of missing elements in the design such as clock trees. By using the **-ct_references** switch, the clock tree estimation step can be customized by specifying specific buffers and/or inverters that would be used in real clock tree synthesis.

In power prediction mode, using **report_power** after **compile_ultra** reports the correlated power. Currently the different switches of **report_power** are not supported.

The power prediction mode setting is saved into the design.

EXAMPLES

The following example sets the power prediction optimization mode and checks out the DC Ultra license.

```
prompt> set_power_prediction
```

Information: Power prediction mode successfully set. (UIO-67)

1

```
prompt> compile_ultra
```

```
prompt> report_power
```

The following example resets the power prediction mode.

```
prompt> set_power_prediction false
```

Information: Power prediction mode successfully reset. (UIO-71)

The following example shows how to account for power of DFT logic.

```
prompt> insert_dft
```

```
prompt> set_power_prediction
```

```
prompt> compile_ultra -incremental
```

```
prompt> report_power
```

SEE ALSO

`compile_ultra(2)`

`report_power(2)`

set_power_switch_cell

Defines the specified library cells as power-switch cells.

SYNTAX

```
status set_power_switch_cell  
  cell_name  
  [-cell_type coarse_grain | fine_grain]  
  [-is_macro]  
  [-switch_pin pin_name]  
  [-pg_pin {pin_name switch_function pg_function}]
```

Data Types

<i>cell_name</i>	string
<i>pin_name</i>	string
<i>switch_function</i>	string
<i>pg_function</i>	string

ARGUMENTS

cell_name

Specifies the name of the library cell to be defined as a power-switch cell.

-cell_type coarse_grain | fine_grain

Specifies the type of the power-switch cell. The type can be either coarse_grain or fine_grain.

-switch_pin *pin_name*

Specifies the switch pin of the power-switch cell.

-pg_pin {*pin_name* *switch_function* *pg_function*}

Specifies the power or ground pin of the power-switch cell.

DESCRIPTION

The **set_power_switch_cell** command sets the specified library cells as power-switch cells.

EXAMPLES

In the following example, the library cells whose names starts with FOOT are set as course-grain power-switch cells.

```
prompt> set_power_switch_cell FOOT* -cell_type coarse_grain \  
-switch_pin SLEEPN -pg_pin {VSS !SLEEPN VSS}
```

SEE ALSO

report_lib(2)
check_library(2)

set_prefer

Sets the **preferred** attribute on the specified library cells.

SYNTAX

```
status set_prefer  
[-min]  
cell_list
```

Data Types

cell_list list

ARGUMENTS

-min

Specifies that the library cells are preferred during hold violation (minimum path) fixing.

cell_list

Specifies the cells on which to set the **preferred** attribute. Cell names must contain a library prefix. If you specify more than one cell name, enclose the list using quotation marks (") or braces ({}).

DESCRIPTION

The **set_prefer** command sets the **preferred** attribute on the specified library cells. The cells must be in one of the target libraries or in a library already loaded into memory. If this attribute exists on a library cell, optimization uses the library cell before other cells in the target library that implement the same function.

Note that the effect of the **preferred** attribute might not be noticeable during optimization because nonpreferred cells can be chosen to help meet constraints.

If you use the **-min** option, the specified library cells are preferred during hold violation fixing. Only buffers and inverters, which are used to fix hold violations, should be included in the cell list. Including cells with any other functionality in the list with the **-min** option has no impact on the optimization engine. You should be careful when using this option, because the preferred cells are used for minimum path fixing, even if it results in a worse area or cell count solution.

If you use the **set_fix_hold_options -preferred_buffer** command together with the **set_prefer -min** command, the **dont_use** attribute on the specified library cell is ignored during hold violation fixing, but is honored by other optimization engines.

The **set_prefer** command affects only the version of the library that is currently loaded into memory and has no effect on the version

that exists on disk. However, if the library is written using the **write_lib** command, the **preferred** attribute is also written, which causes the cells to be permanently preferred.

To remove the **preferred** attribute, use the **remove_attribute** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command specifies that the GA and GB cells from the tech_lib library be preferred during optimization.

```
prompt> set_prefer {tech_lib/GA tech_lib/GB}
```

The following command specifies that the GA and GB cells are preferred during hold fixing (dont_use ignored), and are not used by other optimization engines (dont_use honored).

```
prompt> set_dont_use {tech_lib/GA tech_lib/GB}
prompt> set_prefer -min {tech_lib/GA tech_lib/GB}
prompt> set_fix_hold_options -preferred_buffer
```

SEE ALSO

remove_attribute(2)
report_lib(2)
translate(2)
target_library(3)

set_preferred_routing_direction

Sets the preferred routing direction for the specified routing layers.

SYNTAX

```
string set_preferred_routing_direction  
-layers list_of_layers  
-direction horizontal | vertical
```

ARGUMENTS

-layers *list_of_layers*

Specify the list/collection of layer(s) for which the preferred routing directions is to be set.

-direction horizontal | vertical

Specify the preferred routing direction. The allowed values are: horizontal, vertical, h, v, HORIZONTAL, VERTICAL, H, V. Specify any one.

DESCRIPTION

The `set_preferred_routing_direction` command is used to set the preferred direction for the specified routing layers. This command will be useful in overriding the default layer directions specified in the library itself or even the design. The setting of layer direction done through this command will become specific to this design only.

Note: The command will issue suitable warnings whenever adjacent layers have the same direction after the execution of the command.

Note: This command overrides the preferred direction specified in the reference libraries and saves the user setting as persistent data (will be written when user saves the design) in the design database.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example assigns layers "m1" & "m2" the direction horizontal.

```
prompt> set_preferred_routing_direction \  
       -layers {m1 m2} -direction horizontal
```

SEE ALSO

```
remove_preferred_routing_direction(2)  
report_preferred_routing_direction(2)
```

set_preferred_scenario

Sets the preferred scenario.

SYNTAX

status **set_preferred_scenario**
[*scenario_name*]

Data Types

scenario_name string

ARGUMENTS

scenario_name

Specifies the preferred scenario name. When specified, the scenario must be an active scenario. If not specified, the preferred scenario is reset.

DESCRIPTION

The **set_preferred_scenario** command sets the preferred scenario used for multimode or multicorner optimization. When a scenario is specified as preferred, the scenario is treated as the most constraining scenario. The preferred scenario must be an active scenario or the tool generates an error. When no scenario name is specified, the preferred scenario is reset.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following command set the s1 scenario as the preferred scenario:

```
prompt> set_preferred_scenario s1  
Preferred scenario is set to s1.  
1
```

SEE ALSO

`create_scenario(2)`
`compile_ultra(2)`
`set_active_scenarios(2)`

set_preserve_clock_gate

Sets the **pwr_cg_preservation_type** attribute on clock-gating objects.

SYNTAX

```
status set_preserve_clock_gate  
    cell_collection  
    [-dont_modify_fanout]  
    [-dont_modify_enable]
```

Data Types

cell_collection collection

ARGUMENTS

cell_collection

Specifies the list of cells on which the **pwr_cg_preservation_type** attribute is to be set.

-dont_modify_fanout

Forbids the modification of the fanout for the specified clock gating cells.

-dont_modify_enable

Forbids the modification of the enable function for the specified clock gating cells.

DESCRIPTION

This command sets the **pwr_cg_preservation_type** attribute on clock-gating cells.

The **pwr_cg_preservation_type** attribute ensures a clock-gating element is preserved by the tool, unless there is a conflict with other features. This implies that the tool avoids removing the preserved object. Also, if the tool needs to merge two clock-gating objects (to consolidate fanout) and one of them has the **pwr_cg_preservation_type** attribute, it remains in the design after merging.

You can control which kinds of optimizations are allowed on a clock-gating cell by using the flags *-dont_modify_fanout* and *-dont_modify_enable*.

If the option *-dont_modify_fanout* is used, the tool is not allowed to add, remove, or replace the registers and clock gates that are directly driven by the specified clock-gating cells.

Similarly, if the option `-dont_modify_enable` is used, the tool is not allowed to modify the Boolean function corresponding to the enable signal of the specified clock-gating cells.

The possible values that can be assigned to **pwr_cg_preservation_type** are:

preserve (no flag is used)
dont_modify_fanout (only the `-dont_modify_fanout` flag was used)
dont_modify_enable (only the `-dont_modify_enable` flag was used)
unmodifiable (both flags were used)
unmodifiable_read_only (only set by the **identify_clock_gating** command)

If the **pwr_cg_preservation_type** attribute is set on a non-clock-gating element, it is silently ignored by the tool.

If you want to change the current value of **pwr_cg_preservation_type** of a clock-gating cell, you can use again the **set_preserve_clock_gate** command and assign the needed flags.

Use the **report_attribute** command to verify if a clock-gating element has the **pwr_cg_preservation_type** attribute and which value it contains.

You can alternatively use the **set_attribute** command to set the **pwr_cg_preservation_type** attribute on a clock-gating cell. The attribute can only be set to these values: **preserve**, **dont_modify_fanout**, **dont_modify_enable** and **unmodifiable**.

Use the **remove_attribute** command to remove the **pwr_cg_preservation_type** attribute from a clock-gating cell.

NOTE: The **identify_clock_gating** command automatically sets the **pwr_cg_preservation_type** attribute on the clock-gating cells explicitly identified by using the **-gating_elements** option. You are not allowed to remove the **pwr_cg_preservation_type** attribute if the manually identified clock-gating cells do not have the characteristics of a clock gate that can be inserted by the tool.

EXAMPLES

The following example shows the typical flow for using the **set_preserve_clock_gate** command to preserve a user-instantiated clock-gating object, called `user_cg1`. This assigns the **preserve** value:

```
prompt> read_verilog mapped_design.v
prompt> current_design top
prompt> link
prompt> create_clock clk

prompt> identify_clock_gating
prompt> set_preserve_clock_gate user_cg1
```

The following example shows how to preserve a user-instantiated clock-gating object and not allow the modification of its fanout. This assigns the **dont_modify_fanout** value:

```
prompt> identify_clock_gating
prompt> set_preserve_clock_gate user_cg1 -dont_modify_fanout
```

The following example shows how to preserve a user-instantiated clock-gating object and not allow the modification of its enable function. This assigns the **dont_modify_enable** value:

```
prompt> identify_clock_gating
prompt> set_preserve_clock_gate user_cg1 -dont_modify_enable
```

The following example shows how to preserve a user-instantiated clock-gating object and neither allow the modification of its fanout nor its enable function. This assigns the **unmodifiable** value:

```
prompt> identify_clock_gating  
prompt> set_preserve_clock_gate user_cg1 -dont_modify_fanout -dont_modify_enable
```

SEE ALSO

- identify_clock_gating(2)
- remove_attribute(2)
- report_attribute(2)
- set_attribute(2)
- set_clock_gating_style(2)

set_propagated_clock

Specifies propagated (rather than ideal) clock latency for listed objects.

SYNTAX

```
status set_propagated_clock  
      object_list
```

Data Types

object_list list

ARGUMENTS

object_list

Specifies a list of clocks, ports, or pins on which to apply propagated clocking.

DESCRIPTION

This command specifies that delays be propagated through the clock network to determine latency at register clock pins, including the delay resulting from parasitic capacitance and resistance. If propagated clocking is not specified, the tool uses ideal clocking by default.

Ideal clocking means clock networks have a specified latency (from the **set_clock_latency** command), or zero latency by default. Latency is the amount of time a clock signal takes to be propagated from the ideal waveform origin point to the clock pin of the sequential device.

Propagated clock latency is used for after final clock tree generation and layout. Ideal clock latency specifies a prelayout estimate of the clock tree delay.

If you apply the **set_propagated_clock** command to pins or ports, it affects all register clock pins in the transitive fanout of the specified pins or ports.

To undo the effects of the **set_propagated_clock** command, use the **remove_propagated_clock** command or the **set_clock_latency** command to specify the ideal latency.

To report propagated clock attributes on clocks, use the **report_clock -skew** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

This example specifies the usage of propagated clock latency for all clocks in the design.

```
prompt> set_propagated_clock [all_clocks]
```

SEE ALSO

- `remove_propagated_clock(2)`
- `set_clock_latency(2)`
- `report_clock(2)`
- `set_input_delay(2)`
- `create_clock(2)`

set_qtm_global_parameter

Sets a global parameter for quick timing models (QTM).

SYNTAX

```
status set_qtm_global_parameter
[-param parameter]
[-lib_cell lib_cell]
[-pin pin_name]
[-clock pin_name]
[-value parameter_value]
```

Data Types

<i>parameter</i>	string
<i>lib_cell</i>	string
<i>pin_name</i>	string
<i>parameter_value</i>	float

ARGUMENTS

-param *parameter*

Specifies the global parameter to set. The global parameter can be **setup**, **hold**, or **clk_to_output**.

-lib_cell *lib_cell*

Specifies the *lib_cell* you want to use to mimic the global parameter.

-pin *pin_name*

Specifies the related pin for the global parameter if you are using the **-lib_cell** option to mimic the behavior. If the parameter is **setup** or **hold**, this pin must be an input pin. If the parameter is **clk_to_output**, this pin must be an output pin. If you do not use this option, QTM uses the first constrained pin that it encounters for assigning the **setup** or **hold** parameter. The tool uses the first output pin which has an edge delay arc coming into it for assigning the **clk_to_output** parameter.

-clock *pin_name*

Specifies the constraining pin when you specify the **-param setup** option or the **-param hold** option. This option specifies the launch pin when you specify the **-param clk_to_output** option.

-value *parameter_value*

Specifies the value of the global parameter in time units.

DESCRIPTION

This command sets the value of a desired quick timing model global parameter. The global parameter can be either setup time, hold time, or clk_to_output delay. These parameters specify the global parameter which is added to the timing arcs. The quick timing model setup arc is added to the global setup time to determine the total setup time. The quick timing model hold arc is added to the global hold time to determine the total hold time. The quick timing model edge triggered delay arc is added to the global clk_to_output time to determine the total delay.

You can specify the global parameters of the sequential element by specifying a cell in the library with the **-lib_cell** option. You can also provide the actual value by using the **-value** option. If you specify the **-lib_cell** option, you can specify the clock pin and the input pin name. If you do not specify the clock and input pins, the tool chooses the first setup, hold, or clk_to_output arc that is encountered based on which parameter is being set. If you specify only the **-clock** option, the tool chooses the first setup, hold, or clk_to_output arc (depending on the global parameter) from that clock pin. If you specify only the **-pin** option, the tool chooses the first setup, hold, or clk_to_output arc (depending on the type of global parameter) arriving at the pin.

To display information about the current quick timing model, use the **report_qtm_model** command.

For a basic description of quick timing model, see the **create_qtm_model** man page. For a more detailed description about quick timing model, see the *ICC Design Planning User Guide*.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

Some of the following examples use the **set_qtm_global_parameter** command with the **-lib_cell** option. When you use the **-lib_cell** option, you must also load the technology library and use the **set_qtm_technology** command as shown in the following example:

```
prompt> read_db my_technology_library.db
prompt> set_qtm_technology -library my_technology_library
```

The following example sets the global setup time equivalent to the setup time of the DFF1 cell.

```
prompt> set_qtm_global_parameter -param setup -lib_cell DFF1
```

The following example sets the global hold time equivalent to the setup time of pin D of the DFF1 cell.

```
prompt> set_qtm_global_parameter -param setup -lib_cell DFF1 -pin D
```

The following example sets the global clk_to_out time to be 1.5 time units.

```
prompt> set_qtm_global_parameter -param clk_to_output -value 1.5
```

SEE ALSO

```
create_qtm_constraint_arc(2)
create_qtm_delay_arc(2)
create_qtm_drive_type(2)
create_qtm_load_type(2)
```

```
create_qtm_model(2)
create_qtm_path_type(2)
report_qtm_model(2)
save_qtm_model(2)
```

set_qtm_port_drive

Sets the drive on quick timing model (QTM) ports.

SYNTAX

```
status set_qtm_port_drive
  [-type drive_type]
  [-value drive_value]
  [-input_transition_rise rtrans]
  [-input_transition_fall ftrans]
  port_list
```

Data Types

```
drive_type   string
drive_value  float
rtrans       float
ftrans       float
port_list    collection
```

ARGUMENTS

-type *drive_type*

Specifies the global *drive_type* parameter.

To define the drive types, use the **create_qtm_drive_type** command.

-value *drive_value*

Specifies the drive value in drive resistance units.

-input_transition_rise *rtrans*

Specifies the input rising transition time associated with the input pin of the specified drive type to compute the delay for the drive arc for this port. If you do not include this option, the delay table for the rising input of the drive arc is loaded from the library as-is.

Use the **-input_transition_rise** and **-input_transition_fall** options to capture the transition time associated with the input pin defined for the referenced drive type. This can result in more accurate information on the transition time and delay time for the drive arc defined for this port.

-input_transition_fall *ftrans*

Specifies the input falling transition time associated with the input pin of the specified drive type to compute the delay for the drive arc for this port. If you do not include this option, the delay table for the falling input of the drive arc is loaded from the library as-is.

port_list

Specifies the input or inout ports on which to set the attribute. Each element in the list is either a collection of QTM ports or a pattern that matches QTM ports.

DESCRIPTION

This command sets the drive of an output port of a QTM model. There are two methods for setting the drive:

1. If you have defined drive types, use one of the global *drive_type* parameters.
2. Define the drive in drive resistance units.

To show information about the current QTM model, use the **report_qtm_model** command.

For a basic description about QTMs, see the **create_qtm_model** man page. For a more detailed description about QTMs, see the *IC Compiler Design Planning User Guide*.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following command sets the drive of output port OUT1 to be of drive type drive1.

```
prompt> set_qtm_port_drive -type drive1 OUT1
```

The following command sets the drive of the output port OUT2 to be 5 drive units.

```
prompt> set_qtm_port_drive -value 5.0 OUT2
```

SEE ALSO

create_qtm_drive_type(2)
create_qtm_model(2)
report_qtm_model(2)
save_qtm_model(2)
set_qtm_port_load(2)

set_qtm_port_load

Sets the load on quick timing model (QTM) ports.

SYNTAX

```
status set_qtm_port_load
      [-type load_type]
      [-factor multiplication_factor]
      [-value load_value]
      port_list
```

Data Types

<i>load_type</i>	string
<i>multiplication_factor</i>	float
<i>load_value</i>	float
<i>port_list</i>	collection

ARGUMENTS

-type *load_type*

Specifies the global **load_type** parameter.

To define the load types, use the **create_qtm_load_type** command.

-factor *multiplication_factor*

Specifies the multiplication factor for the load type to set the load on the QTM port.

-value *load_value*

Specifies the load value in the capacitance units.

port_list

Specifies the input or inout ports on which to set the attribute. Each element in the list is either a collection of QTM ports or a pattern that matches QTM ports.

DESCRIPTION

This command sets the load of an input port of a QTM model. There are two methods for setting the load:

1. If you have load types defined, use the global *load_type* parameter along with a multiplication factor.
2. Define the load in terms of the capacitance units.

To show information about the current QTM model, use the **report_qtm_model** command.

For a basic description about QTMs, see the **create_qtm_model** man page. For a more detailed description about QTMs, see the *IC Compiler Design Planning User Guide*.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following command sets the load on IN1 to be 2 times the global *load_type* load1.

```
prompt> set_qtm_port_load -type load1 -factor 2 IN1
```

The following command sets the load on IN2 to be 4.5 capacitance units.

```
prompt> set_qtm_port_load -value 4.5 IN2
```

SEE ALSO

create_qtm_load_type(2)
create_qtm_model(2)
report_qtm_model(2)
save_qtm_model(2)
set_qtm_port_drive(2)

set_qtm_technology

Sets quick timing model technology variables.

SYNTAX

```
string set_qtm_technology  
  [-library name]  
  [-max_transition max_trans_value]  
  [-min_transition min_trans_value]  
  [-max_capacitance max_cap_value]  
  [-min_capacitance min_cap_value]  
  [-wire_load_model wlm_name]  
  [-operating_condition opcond_name]  
  [-process process_value]  
  [-voltage voltage_value]  
  [-temperature temperature_value]
```

Data Types

<i>name</i>	string
<i>max_trans_value</i>	float
<i>min_trans_value</i>	float
<i>max_cap_value</i>	float
<i>min_cap_value</i>	float
<i>wlm_name</i>	string
<i>opcond_name</i>	string
<i>process_value</i>	float
<i>voltage_value</i>	float
<i>temperature_value</i>	float

ARGUMENTS

-library *name*

Specifies the library name for using library elements to define global parameters. The library is the timing library in the Synopsys database (.db) file, not the Milkyway reference or design library. Substitute the string value you want for *name*.

-max_transition *max_trans_value*

Specifies the maximum transition value to use. You must express it in units consistent with the those in the technology library used during optimization. For example, if the library specifies drive values in kilohms and load values in picofarads, then you must express *max_transition_value* in nanoseconds. Substitute the float value you want for *max_trans_value*.

-min_transition *min_trans_value*

Specifies the minimum transition value to use. You must express it in units consistent with the those in the technology library used during optimization. For example, if the library specifies drive values in kilohms and load values in picofarads, then you must express *min_transition_value* in nanoseconds. Substitute the float value you want for *min_trans_value*.

-max_capacitance *max_cap_value*

Specifies the maximum value to use for capacitance. You must express it in units consistent with the technology library used during optimization. For example, if the library specifies capacitance values in picofarads, then you must express *max_cap_value* in picofarads. Substitute the float value you want for *max_cap_value*.

-min_capacitance *min_cap_value*

Specifies the minimum value to use for capacitance. You must express it in units consistent with the technology library used during optimization. For example, if the library specifies capacitance values in picofarads, then you must express *min_cap_value* in picofarads. Substitute the float value you want for *min_cap_value*.

-wire_load_model *wlm_name*

Specifies the wire load model to use when calculating delays. Substitute the string value you want for *wlm_name*.

-operating_condition *opcond_name*

Specifies the default operating condition name for the quick timing model in a multicorn, multimode (MCMM) design. Must be specified together with -process, -voltage, and -temperature.

-process *process_value*

Specifies the process scaling factor for the default operating condition of the quick timing model in an MCMM design. When you want to use any of the following options, you must use all of them: **-operating_condition**, **-process**, **-voltage**, and **-temperature**. Allowed values are 0.0 through 100.0.

-voltage *voltage_value*

Specifies the voltage value, in Volts, for the default operating condition of the quick timing model in an MCMM design. When you want to use any of the following options, you must use all of them: **-operating_condition**, **-process**, **-voltage**, and **-temperature**.

-temperature *temperature_value*

Specifies the temperature value, in degrees Celsius, for the default operating condition of the quick timing model in an MCMM design. When you want to use any of the following options, you must use all of them: **-operating_condition**, **-process**, **-voltage**, and **-temperature**.

DESCRIPTION

This command sets various quick timing model technology parameters. You can use the **-library** option to define path types, drive types, and load types, or if you set the other global parameters in terms of library cells. Before setting the library, you ensure that it is loaded.

If you have set the wire load model, the path type delays are calculated by using that wire load model.

You can use the **report_qtm_model** command to show information about the current quick timing model.

For a basic description of quick timing models, see the **create_qtm_model** man page. For a more detailed description of quick timing models, see *Creating Quick Timing Models for Black Boxes* in *IC Compiler User Guide: Design Planning Contents*.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example sets the library to the name my_lib.

```
prompt> read_lib my_lib.db  
#my_lib.db contain the library my_lib  
prompt> set_qtm_technology -library my_lib
```

The following command sets the wire_load_model to wlm1.

```
prompt> set_qtm_technology -wire_load_model wlm1
```

SEE ALSO

```
create_qtm_model(2)  
report_qtm_model(2)  
save_qtm_model(2)
```

set_query_rules

Defines rules for rule-based query.

SYNTAX

```
status set_query_rules
[-hierarchical_separators separator_list]
[-bus_name_notations bus_name_list]
[-class class_list]
[-regsub regsub]
[-regsub_cumulative]
[-wildcard]
[-suffix suffix_list]
[-verbose]
[-nocase]
[-reset]
[-show]
```

Data Types

```
separator_list  list
bus_name_list  list
class_list     list
regsub         list
suffix_list    list
```

ARGUMENTS

-hierarchical_separators *separator_list*

Specifies a list of equivalent hierarchy separators in object names. There must be at least two elements in the list. The default list is {/ _ .}. The elements are preferably all single character, although at most one multi-character element is allowed. No more than one alphabetical character a-z or A-Z is allowed as a hierarchical separator. The single alphabetical character element is also mutually-exclusive with the multi-character element. The following special characters are not supported as hierarchy separators nor they are allowed as one of the characters for the multi-character hierarchy separator: 0-9, '*', '?', '\', '+', '^', '[', ']', '(', ')', '<', '>', and '{', '}'.

-bus_name_notations *bus_name_list*

Specifies a list of equivalent bus name notations in object names. There must be at least two elements in the list, and the list element must be of two characters exactly. The first character of the element is the opening bus name character, and the second character is the closing bus name character. The default list is {[_ ()]}. No more than one matching pair of alphabetical characters a-z or A-Z is allowed as bus name notations. The following characters are not supported as opening or closing bus name characters: 0-9, '*', '?', '\', '+', '^', and '/. Bracketed bus notations must be paired. For example, "[)" are not properly paired

brackets; therefore, it is not supported. For non-bracket bus name notations, the opening and closing bus name character must be the same. For example, "-_" is not a supported bus name notation.

-class *class_list*

Specifies a list of object class names for which the query rules are applied. The following object classes are supported: cell, port, pin, net, power_domain, supply_net, supply_set, supply_port, and power_switch. By default, they are all enabled; however, you can specify a subset of them.

-regsub *regsub*

Specifies a list of options for the **regsub** Tcl command. Each list should include three string elements {{switches} {match_regexp} {substitution_exp}}. The query uses the **-regsub** option to evaluate the object name and pattern strings during the matching process.

The **-regsub** option can be specified multiple times in one **set_query_rules** command. When multiple **-regsub** options are used, the **-regsub** options are applied to the leaf object name and pattern in the order you specify.

The **-regsub** option is only applicable to pin, port, net, and leaf cells. If **-regsub** is used without other query rules, the default values for the other options are still used. To save runtime, use this option only if the existing rule-based matching scheme fails to match an object.

In addition to the Tcl built-in options for the **regsub** command, the following options are supported:

- **-class** specifies a single object class name, such as cell, port, pin, or net. The **-class** option makes the **-regsub** query rule object specific. The object class specific **-regsub** rule is preferred to the legacy global **-regsub** rule. The latter has been deprecated. For backward compatibility purposes, the global **-regsub** query rule is automatically converted to the class specific rules for the port, pin, and net classes, but not for the cell class.
- **-cumulative** specifies that the multiple **-regsub** rules for the same object class are executed with cumulative effect so that the subsequent **-regsub** rule is applied to the outputs of the object name and pattern string from the previous **-regsub** rules. After this option is used for an object class with the **-regsub** option, all the **-regsub** rules for that object class will have the cumulative option. The legacy global switch **-regsub_cumulative** is deprecated and is automatically converted to the class specific **-cumulative** option for the port, pin, and net classes, but not for the cell class.

-regsub_cumulative

This option is deprecated. For backward compatibility, it is automatically converted to the class specific **-cumulative** option for the port, pin, and net classes, but not for the cell class.

-wildcard

Enables wildcard support in rule-based matching. Be aware that this option might match more objects after ungrouping and **change_names**.

-suffix *suffix_list*

Enables the suffix name rule. Only the predefined suffix *_reg* and *_tri* are supported. Suffix *_reg* is only applicable to register cells. When enabled, a cell name pattern such as "U4" would match register cell named "U4_reg". Suffix *_tri* is only applicable to tristate cells. When enabled, a cell name pattern such as "U8" would match tristate cell object named "U8_tri".

-verbose

Prints messages indicating the object is matched using query rules.

-nocase

Enables case-insensitive rule-based matching. This is the preferred way of enabling case-insensitive rule-based matching.

-reset

Resets query rules to the program default.

-show

Shows the current definition of the query rules. If used with other options, the new query rules will be set first and then displayed.

DESCRIPTION

The **set_query_rules** command allows you to define query rules for some applications, such as **extract_physical_constraints** to guide the object queries with query rules. The purpose of the rule-based matching is to resolve the naming differences that are often due to ungrouping and **change_names** command activity. The hierarchy separators and bus names are typical sources of the naming differences.

When a list of characters is defined as equivalent in terms of hierarchy separator, the object query uses the equivalence when matching the objects in the in-memory netlist. For example, if the list {/_ .} is defined as equivalent hierarchy separators, the cell named "a.b_c/d_e" is matched with name string "a/b_c.d/e" provided in the Tcl scripts or in the DEF files.

When a list of bus notations is defined as equivalent in terms of bus names, the object query uses the equivalence when matching the objects in the in-memory netlist. For example, if the list {[] _} is defined as equivalent bus notations, the cell named "a[4][5]" is matched with the name string "a_4__5_" provided in the Tcl scripts or DEF files.

When the **-regsub** rules are specified, the Tcl **regsub** command applies the rules to both the object name and the pattern string for rule-based matching if the other rule-based matching scheme fails to match an object.

Be cautious about enabling wildcard support in a rule-based match because the wildcard match might give you more matches than you want.

Use the **-reset** option to reset the query rules to the program default. Use **-show** to report the current query rules. When **-show** is used with other options, the new rules are set first and then reported.

The **-nocase** option of command **set_query_rules** is the preferred way of enabling case-insensitive rule-based matching, although global variable **find_ignore_case** variable is honored. For runtime reasons, avoid using the **-nocase** option in query commands such as **get_cells** or **get_pins**.

The **set_query_rules** command can be issued multiple times. Unless the **-show** option is used alone, the new rules always overwrite the previous rule settings.

Be cautious when defining and enabling query rules. To save runtime, do not use unnecessary query rules.

Rule-based matching is enabled by the **enable_rule_based_query** variable. The default or user-defined query rules are effective for object queries only when the **enable_rule_based_query** variable is set to **true**. Runtime for rule-based matching could be much slower. Rule-based matching should be disabled once it is no longer needed. By default, rule-based matching is off.

EXAMPLES

```
# Show program default.
prompt> set_query_rules -show
```

```
*****
Query Rules
*****
```

```

      Rule Type  Rule Value
      -----  -
Hierarchy-Separator {/_ _}
Bus-Notation  {[] __ ()}
Object-Class  {cell port pin net power_domain power_switch supply_net supply_set supply_port}
Nocase       false
Wildcard     false
Verbose      false

```

1

Non-default

```

prompt> set_query_rules \
  -hierarchical_separators {/_ _} \
  -bus_name_notations {[] ||} \
  -class {cell port} -show

```

Query Rules

```

      Rule Type  Rule Value
      -----  -
Hierarchy-Separator {/_ _}
Bus-Notation  {[] ||}
Object-Class  {cell port}
Nocase       false
Wildcard     false
Verbose      false

```

1

Use alphabetical characters

```

prompt> set_query_rules \
  -hierarchical_separators {/_ x} \
  -class {cell port} \
  -bus_name_notations {[] __ || xx} -show

```

Query Rules

```

      Rule Type  Rule Value
      -----  -
Hierarchy-Separator {/_ x}
Bus-Notation  {[] __ || xx}
Object-Class  {cell port}
Nocase       false
Wildcard     false
Verbose      false

```

1

Unsupported usage

```

prompt> set_query_rules \
  -hierarchical_separators {/_ x y} \
  -bus_name_notations {[] __ || xx}

```

Error: At most 1 alphabetical or multi-character hierarchy separator is allowed. (UID-1067)

0

Setting regsub rules

```
prompt> set_query_rules -show \
-class {port cell pin net} \
-regsub {{-class cell} {(.*)(\d+)} {1_2} } \
-regsub {{-nocase -class net -cumulative} {_BAR$} {}} \
-regsub {{-class net} {RED} {3}}
```

Query Rules

Rule Type	Rule Value
Hierarchy-Separator	{/_ .}
Bus-Notation	{[] __ ()}
Object-Class	{cell port pin net}
regsub	{{-class cell} {(.*)(\d+)} {1_2}}
regsub	{{-class net -cumulative -nocase} {_BAR\$} {}}
regsub	{{-class net -cumulative} {RED} {3}}
Nocase	false
Wildcard	false
Verbose	false

1

Automatic conversion of deapreciated global regsub rules

```
prompt> set_query_rules -show \
-class {port cell pin net}
-regsub {{-class cell} {(.*)(\d+)} {1_2} } \
-regsub {{-nocase} {_BAR$} {}} \
-regsub {{} {RED} {3}}
```

Information: Global options '-regsub' and '-regsub_cumulative' are obsolete. They are converted to port, pin and net regsub rules. (l

Information: Global options '-regsub' and '-regsub_cumulative' are obsolete. They are converted to port, pin and net regsub rules. (l

Query Rules

Rule Type	Rule Value
Hierarchy-Separator	{/_ .}
Bus-Notation	{[] __ ()}
Object-Class	{cell port pin net}
regsub	{{-class cell} {(.*)(\d+)} {1_2}}
regsub	{{-class port -nocase} {_BAR\$} {}}
regsub	{{-class port} {RED} {3}}
regsub	{{-class pin -nocase} {_BAR\$} {}}
regsub	{{-class pin} {RED} {3}}
regsub	{{-class net -nocase} {_BAR\$} {}}
regsub	{{-class net} {RED} {3}}
Nocase	false
Wildcard	false
Verbose	false

1

Rule-based query examples

```

prompt> set enable_rule_based_query true
Information: Rule-based matching enabled. Please turn it off once rule-based matching is no longer needed. (UID-1056)
true
prompt> set_query_rules \
-hierarchical_separators {/_ .} \
-bus_name_notations {[ ] __}
1
prompt> get_cells b.in_a_c.in_b/d1_in.c/U.in_d
{b.in_a/c.in_b/d1/in_c/U.in_d}
1
prompt> set_query_rules \
-hierarchical_separators {/_ . _H_} \
-bus_name_notations {[ ] __ ||}
1
prompt> get_cells b.in_a.c_H_in_b_d1_H_in.c_3__H_a|4|[5]/d_3/f_4_d_H_z|2|
{b.in_a/c.in_b/d1_in.c_3_/a_4__5_/d_3/f_4_d/z_2_}
1
prompt> set_query_rules -suffix {_reg_tri} -wildcard
1
prompt> get_cells {fr2/regout[*]}
{fr2_regout_reg_0_fr2_regout_reg_1_fr2_regout_reg_2_fr2_regout_reg_3_}
1
prompt> get_pins {fr2_regout[*2*]/C*}
{fr2_regout_reg_2_/CLK}
1
prompt> set_query_rules \
-regsub {{-class cell} XYZ_2_reg_3_} -wildcard
1
prompt> get_cells fr2/regout_XYZ_2_
{fr2_regout_reg_3_}
1
prompt> get_pins fr2.regout_XYZ_2_/C*K
{fr2_regout_reg_3_/CLK}
1
prompt> set_query_rules \
-regsub {{-class net} {_BAR$} {}} \
-regsub {{-class net -cumulative} {RED} {3}}
1
prompt> get_nets {pn/a[RED]}
{pn_a_3__BAR}
1
prompt> set enable_rule_based_query false
false

```

SEE ALSO

```

enable_rule_based_query(3)
find_ignore_case(3)
regsub(2)
extract_physical_constraints(2)
get_cells(2)
get_pins(2)

```


get_nets(2)

set_register_merging

Sets the **register_merging** attribute on the specified cells or designs, allowing register merging optimization on the objects.

SYNTAX

```
status set_register_merging  
  obj_list  
  [true | false]
```

Data Types

obj_list list

ARGUMENTS

obj_list

Specifies a list of cell or design names for which to enable register merging optimization. Cell names in *obj_list* must be from the current design. If multiple objects are specified, they must be enclosed in quotation marks (") or braces ({}).

true | false

Specifies the value with which to set the **register_merging** attribute. When this option is set to **true** (the default), register merging is enabled on the specified objects (cells or designs). Register merging is performed on the specified objects provided that the **compile_enable_register_merging** variable is set to **true**.

Setting **false** on a design disables register merging on the whole design. Setting **false** on an instance disables register merging on the whole instance.

DESCRIPTION

The **set_register_merging** command sets the **register_merging** attribute on the specified *obj_list*. This attribute is used to specify the cells or designs to be optimized using register merging.

If a cell with a specified name is found in the current design, the **register_merging** attribute is set to the specified value in the cell. If no cell with a specified name is found, the tool searches for a design and the attribute set for the design.

EXAMPLES

The following example shows how to enable register merging optimization during optimization for the sequential cells named U0 and U1 and how to disable register merging for the cell named U2:

```
prompt> set_register_merging {U0 U1} TRUE  
prompt> set_register_merging U2 FALSE
```

SEE ALSO

- `compile(2)`
- `get_attribute(2)`
- `remove_attribute(2)`
- `compile_enable_register_merging(3)`

set_register_replication

Sets the **register_replication** attribute on the specified sequential cells, thus allowing register replication on the objects.

SYNTAX

```
int set_register_replication  
  [-max_fanout max_fanout_value]  
  [-num_copies copy_value]  
  [-include_fanin_logic cell]  
  [-include_fanout_logic cell]  
  [-driven_by_original_register end_points_list]  
  object_list
```

Data Types

```
max_fanout_value  integer  
copy_value       integer  
cell             string  
end_points_list  list  
object_list      list
```

ARGUMENTS

-max_fanout *max_fanout_value*

Specifies the maximum fanout of the register after the replication.

-num_copies *copy_value*

Specifies the number of copies for the specified register is replicated.

-include_fanin_logic *cell*

Specifies the combinational cell in the fanin of the register for a path replication.

-include_fanout_logic *cell*

Specifies the combinational cell in the fanout of the register for a path replication.

-driven_by_original_register *end_points_list*

Specifies the endpoints cells (registers or primary output ports) that should remain connected to the original register.

object_list

Specifies the set of registers to replicate.

DESCRIPTION

Sets the **register_replication** attribute on *object_list*. This attribute is used to specify how many copies of the sequential cells are replicated. When you specify the **-num_copies** option, the **register_replication** attribute value is the value of the **-num_copies** option. Otherwise, it is the value of the **-max_fanout** option.

The **-include_fanin_logic** option can be used to replicate a path. A single object is specified for the option and denotes the start of the path. Only a single object can be specified in the **object_list** when **-include_fanin_logic** option is used.

The **-include_fanout_logic** option can be used to replicate paths from a set of registers to a specific cell (choke point). A single object is specified for the option and denotes the end of the paths. Multiple registers can be specified in the **object_list** when **-include_fanout_logic** option is used. There must be a unique path from each register of the set to the choke point cell. The replication would be completed as a single action for the given registers.

The **-driven_by_original_register** option can be used to keep specified endpoints cells connected to the original register when replicating registers.

If a sequential cell with a specified name is found in the *current design*, the **register_replication** attribute is set to the specified value in the cell.

EXAMPLES

The following example shows how to replicate sequential cell U0 three times and how to replicate sequential cell U1 so that its fanout is less than or equal to four.

```
prompt> set_register_replication -num_copies 3 U0
prompt> set_register_replication -max_fanout 4 U1
```

The following example shows how to get an additional copy of the path from the U1 cell to the A_reg register, which lies in the fanin of the register:

```
prompt> set_register_replication -num_copies 2 A_reg -include_fanin_logic U1
```

The following example shows how to get an additional copy of the paths from A_reg B_reg registers to the choke point cell U1, which lies in the fanout of the registers:

```
prompt> set_register_replication -num_copies 2 {A_reg B_reg} -include_fanout_logic U1
```

The following example shows how to use the **-driven_by_original_register** option. The r3_reg, r2_reg, and r1_reg registers are in the transitive fanout of the reg_1_reg original register. The 3 registers remain connected to the original register after replication.

```
prompt> set_register_replication -num_copies 2 reg_1_reg \
-driven_by_original_register {r3_reg r2_reg r1_reg}
```

The following example shows how to use the **-driven_by_original_register** option. The op_1 and op_2 output ports are in the transitive fanout of the reg_1_reg original register. The 2 output ports remain connected to the original register after replication.

```
prompt> set_register_replication -num_copies 2 reg_1_reg \
-driven_by_original_register {op_1 op_2}
```

SEE ALSO

- compile(2)
- find(2)
- get_attribute(2)
- remove_attribute(2)

set_register_type

Sets the **latch_type** or **flip_flop_type** attributes on designs or cell instances, to specify which sequential cells from the target library are to be used by the **compile** command.

SYNTAX

```
int set_register_type  
  -latch example_latch -exact |  
  -flip_flop example_flip_flop [-exact]  
  [cell_or_design_list]
```

Data Types

<i>example_latch</i>	string
<i>example_flip_flop</i>	string
<i>cell_or_design_list</i>	list

ARGUMENTS

-latch *example_latch*

Specifies a latch to be used by the **compile** command as the default latch type.

You can specify both **-latch** and **-flip-flop**, but you must specify at least one.

-exact

Instructs the **compile** command to make an exact mapping to the specified *example_latch* or *example_flip_flop*, if possible.

This argument is required when using the **-latch** argument.

-flip_flop *example_flip_flop*

Specifies a flip-flop from the target library to be used by the **compile** command as the default flip-flop type. the *example_flip_flop*, if possible. This argument sets the **default_flip_flop_type** or **default_flip_flop_type_exact** attribute to the *example_flip_flop* on all designs in *cell_or_design_list*; and the **flip_flop_type** or **flip_flop_type_exact** attribute to the *example_flip_flop* on all cells in *cell_or_design_list*.

You can specify both **-latch** and **-flip-flop**, but you must specify at least one.

cell_or_design_list

Specifies a list of cells or designs in which to use the latch or flip-flop. The default is the current design.

DESCRIPTION

Specifies latch or flip-flop type information for the **compile** command to use, by setting appropriate attributes on the designs or cell instances. For designs, specifying **-flip_flop** without **-exact**, sets the **default_flip_flop_type** attribute to *example_flip_flop*, indicating to use it as the default flip-flop type for those designs. And specifying **-exact -flip_flop** sets the **default_flip_flop_type_exact** attribute to *example_flip_flop*, indicating to use it as the exact default flip-flop for those designs.

For example, to set the default flip-flop type for a design to be a D flip-flop, specify the name of any D flip-flop in the target library as the *example_flip_flop*, without **-exact**. If you specify **-exact**, the *example_flip_flop* is interpreted as the exact flip-flop to use.

Similarly, for cell instances, specifying **-flip_flop** without **-exact** sets the **flip_flop_type** attribute to *example_flip_flop*, indicating that it is to be used as the specific flip-flop type for those cells. And specifying **-exact -flip_flop** sets the **flip_flop_type_exact** attribute to *example_flip_flop*, indicating that it is to be used as the exact flip-flop for those cells.

When specifying a latch, you must always use the **-exact** option, so the *example_latch* is always the exact latch used as the default latch for designs and as the specific latch for cells. The **default_latch_type_exact** attribute is set to the specified latch on designs, and the **latch_type_exact** attribute is set to the specified latch on cells.

There are no **latch_type** or **default_latch_type** attributes.

The mapping process for flip-flops and latches consists of two steps:

1. Sequential gates are mapped to an initial latch or flip-flop from the target library. The **compile** command attempts to convert flip-flops or latches tagged by **set_register_type** to the specified flip-flop or latch type. Untagged sequential components are mapped to the default latch or flip-flop type if specified. In the absence of attributes on a design, or a sequential cell instance to control the mapping, the smallest area-cost flip-flop or latch is chosen.
2. If a flip-flop is not specified with the **-exact** option, the tool attempts to individually remap it to a lower-cost component from the target library. If **compile** cannot use the sequential component specified, or if no default flip-flop or latch is specified, **compile** maps these cells into the smallest sequential component possible.

A cell instance can have either latch or flip-flop attributes, but not both. Only latch cells can have latch attributes, and only flip-flop cells can have flip-flop attributes. Designs can have both latch and flip-flop attributes, because these attributes designate the default latch type (for latches in the design) and the default flip-flop type (for flip-flops in the design).

Unmapped sequential components can result when the **translate** command cannot find a close match for a flip-flop or latch in the new target library, or when latches or flip-flops are added to a design by HDL Compiler, VHDL Compiler, or state-machine software.

The **set_register_type** command constrains the mapping process. It does not constrain scan replacement in **insert_dft** or in **compile**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the default flip-flop type for the current design to a D flip-flop, and that one example of a D flip-flop in the current library is the component *DFLOP*. This sets the **default_flip_flop_type** attribute to *DFLOP* on the current design. As a result, the default flip-flop used in mapping components that are not otherwise tagged is a D flip-flop.

```
prompt> set_register_type -flip_flop DFLOP
```

The following example sets the default flip-flop type for the current design exactly to the *DFLOP* cell. This sets the **default_flip_flop_type_exact** attribute to *DFLOP* on the current design. As a result, the default flip-flop used in mapping

components that are not otherwise tagged is DFLOP.

```
prompt> set_register_type -exact -flip_flop DFLOP
```

The following example sets the default latch type for *FOO* and *BAR* designs to exactly a *DLATCH* gate. The **compile** command attempts to map all unmapped latches in these designs to *DLATCH*. This sets the **default_latch_type_exact** attribute to *DLATCH* on the *FOO* and *BAR* designs. The presence of this attribute indicates that an exact mapping is to be attempted for the *DLATCH* latch on the untagged cells in the *FOO* and *BAR* designs. Sequential types for latches must always be set as exact.

```
prompt> set_register_type -exact -latch DLATCH {FOO, BAR}
```

The following example maps a group of cell instances exactly to the *DLATCH* latch. This sets the **latch_type_exact** attribute on the *U1*, *U2*, and *U3* cell instances to the name *DLATCH*.

```
prompt> set_register_type -exact -latch DLATCH {U1, U2, U3}
```

The following example specifies the exact default latch type for the current design as *DLATCH* and the exact default flip-flop type as *DFLOP*. This sets the **default_latch_type_exact** attribute to the name *DLATCH* and the **default_flip_flop_type_exact** attribute to the name *DFLOP* on the current design so that **compile** can map otherwise untagged components in the design. You must use **exact** when you specify both **-latch** and **-flip_flop** in the same command.

```
prompt> set_register_type -exact -latch DLATCH -flip_flop DFLOP
```

SEE ALSO

- current_design(2)
- get_attribute(2)
- remove_attribute(2)
- reset_design(2)
- translate(2)
- attributes(3)
- target_library(3)

set_related_supply_net

Associates a supply net to the port of the design or the pin of a cell.

SYNTAX

```
status set_related_supply_net
  [supply_net_name]
  [-object_list objects]
  [-reset]
  [-ground ground_net_name]
  [-power power_net_name]
```

Data Types

<i>supply_net_name</i>	string
<i>objects</i>	list
<i>ground_net_name</i>	string
<i>power_net_name</i>	string

ARGUMENTS

supply_net_name

Specifies the name of the power net. This is deprecated and supported for backward compatibility only. New scripts should specify the power net using the **-power** option.

-object_list *objects*

Specifies the list of ports (pins) to be associated with power/ground nets or reset. If this option is not specified, all the ports are considered.

-reset

Resets the related power and ground nets of the ports (pins) specified in *object_list*. This cannot be used with the **-power** or **-ground** options.

-ground *ground_net_name*

Specifies the name of the ground net.

-power *power_net_name*

Specifies the name of the power net.

DESCRIPTION

This command is used to associate supply nets with design ports or leaf pins. The tool uses this information for constraining and checking the design. The level-shifter insertion tool also uses supply nets to determine if level shifter is required between a port (pin) and the logic connected to the port (pin).

The supply net is associated with the input ports only if the supply voltage (if already specified for the supply using the **set_voltage** command) matches with the voltage of the driving cell (as specified by the **set_driving_cell** command for the input port). The command issues an error when the voltages do not match. The command also neglects ports that do not meet the voltage matching criteria.

This command cannot be used on ports that have driver or receiver supply port attributes. The driver and receiver supply port attributes have higher precedence than the **related_supply_net** attribute; therefore, this command will not set the **related_supply_net** attribute on such ports and issues an information message.

If this command is not specified, the tool assumes that ports are externally connected to the primary supply net of the domain of the top design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the VDD_EXT power net and the VSS_EXT ground net with the INPUT port.

```
prompt> set_related_supply_net -object_list [get_ports INPUT] \  
-power VDD_EXT -ground VSS_EXT
```

The following example resets the power net and ground net on all ports.

```
prompt> set_related_supply_net -reset
```

SEE ALSO

get_ports(2)
set_port_attributes(2)

set_repeater

Defines the UPF repeater strategy for the power domains in the design.

SYNTAX

```
status set_repeater  
    repeater_strategy_name  
    -domain power_domain  
    [-repeater_supply repeater_supply_set]  
    [-applies_to inputs | outputs | both]  
    [-elements objects]  
    [-exclude_elements exclude_objects]  
    [-name_prefix prefix]  
    [-name_suffix suffix]  
    [-update]
```

Data Types

<i>repeater_strategy_name</i>	string
<i>power_domain</i>	string
<i>repeater_supply_set</i>	string
<i>objects</i>	list
<i>exclude_objects</i>	list
<i>prefix</i>	string
<i>suffix</i>	string

ARGUMENTS

repeater_strategy_name

Specifies the name of the UPF repeater strategy. The name of the repeater strategy should be unique within the specified power domain.

-domain *power_domain*

Specifies the name of the power domain to which this UPF repeater strategy will be applied.

-repeater_supply *repeater_supply_set*

Specifies the supply set whose power and ground functions are to be used as the repeater power and ground nets respectively.

-applies_to inputs | outputs | both

Specifies whether the given **set_repeater** command applies to all inputs or outputs or both types of ports of the power domain.

-elements *objects*

Specifies the objects to which this UPF repeater strategy will be applied. The objects can be pins of the root cells of the power domain or top level ports or root cells of the power domain.

-exclude_elements *exclude_objects*

Specifies the objects to which this UPF repeater strategy won't be applied. The objects can be pins of the root cells of the power domain or top level ports or root cells of the power domain.

-name_prefix *prefix*

Specifies the prefix to use for the repeater cell names.

-name_suffix *suffix*

Specifies the suffix to use for the repeater cell names.

-update

Indicates that this command adds *elements* or *exclude_elements* for a previous command with the same *strategy_name* and *domain_name* and executed in the same scope.

With the **-update** option, the command only accepts **-elements** or **-excluded_elements**, and the required option *strategy_name* and **-domain**. If other options are specified with **-update**, the command will error out.

DESCRIPTION

This command defines the UPF repeater strategy for the ports of the specified power domain.

If the **-elements** and **-applies_to** options are not specified, then the repeater strategy will be applied to all ports of the power domain.

If **-repeater_supply** is not specified, then the command will error out.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to define a UPF repeater strategy for all output ports of the specified power domain PD1.

```
prompt> set_repeater repeater_1 -domain PD1 -repeater_supply PD1.primary -applies_to output\
```

The following example shows how to define a UPF repeater strategy for specific ports of the specified power domain:

```
prompt> set_repeater repeater_2 -domain PD1 \
-repeater_supply rptr_supply_set \
-elements inst/special_port
```

The following example shows how to define a UPF repeater strategy using **-exclude_elements**:

```
prompt> set_repeater repeater_3 -domain PD1 \
-repeater_supply rptr_supply_set \
```

```
-elements inst \  
-exclude_elements inst/special_port
```

SEE ALSO

set_port_attributes(2)

set_replace_clock_gates

Set directives for clock gate replacement. Forces the enabling or disabling of clock gate replacement for specified combinational cells in the current design. Also sets the edge type for modules or black-box cells that otherwise could not be replaced. The cells are replaced by executing the **replace_clock_gates** command.

SYNTAX

```
int set_replace_clock_gates  
  [-include_cells cell_list]  
  [-exclude_cells cell_list]  
  [-rising_edge_clock pin_list]  
  [-falling_edge_clock pin_list]  
  [-undo object_list]
```

Data Types

```
cell_list    list  
pin_list     list  
object_list list
```

ARGUMENTS

-include_cells *cell_list*

Specifies a list of cells in the current design to be included for clock gate replacement. A cell cannot be on more than one of the **-include_cells**, **-exclude_cells**, and **-undo** lists.

-exclude_cells *cell_list*

Specifies a list of cells in the current design to be excluded from clock gate replacement. A cell cannot be on more than one of the **-include_cells**, **-exclude_cells**, and **-undo** lists.

-rising_edge_clock *pin_list*

Specifies a list of pins or ports in the current design to be recognized as a positive edge triggered clock pin during clock gate replacement. A (corresponding) port cannot be on more than one of the **-rising_edge_clock**, **-falling_edge_clock**, and **-undo** lists.

-falling_edge_clock *pin_list*

Specifies a list of pins or ports in the current design to be recognized as a negative edge triggered clock pin during clock gate replacement. A (corresponding) port cannot be on more than one of the **-rising_edge_clock**, **-falling_edge_clock**, and **-undo** lists.

-undo *object_list*

Specifies a list of pins, ports or cells in the current design for which the previous specification to **set_replace_clock_gates** is to be undone. A cell cannot be on more than one of the **-include_cells**, **-exclude_cells**, and **-undo** lists, while a (corresponding) port cannot be on more than one of the **-rising_edge_clock**, **-falling_edge_clock**, and **-undo** lists.

DESCRIPTION

This command specifies combinational cells for which clock gate replacement is to be either enabled or disabled. By default, all combinational cells on the clock paths that are not buffers or inverters are considered for clock gate replacement. Cells on the **-exclude** lists will not be replaced with Power Compiler clock-gating cells, regardless of any previous specifications. If cells are specified on the **-include** lists, only those cells are considered for replacement.

This command is also used to specify the edge type of clock ports for modules or black-box cells that otherwise could not be clock gated by applying clock gate replacement to manually inserted clock gating cells. clock gated otherwise. One of the conditions for clock gate replacement is that the fanout of the manually inserted clock gate should drive only registers that are triggered by the same clock edge. If the fanout contains a black-box cell or sequential cell for which the edge type cannot be identified, or if the edge type is not compatible with the registers in the fanout of the gating cell, it is possible to override the edge detection mechanism used by **replace_clock_gates**.

The same port cannot be specified as **rising_edge** and **falling_edge** at the same time. If an instance pin is specified, the corresponding design port is used instead, since the edge type property is design specific.

You must run **set_replace_clock_edges** before issuing **\replace_clock_gates**; the specifications persist during all subsequent executions of **replace_clock_gates**. Use the **-undo** option to remove the effect of an earlier invocation of **set_replace_clock_gates**.

EXAMPLES

The following example excludes the manually inserted clock-gating cells named **C12** and **C13** in the current design. All other cells are still considered for clock gate replacement.

```
prompt> set_replace_clock_gates -exclude_cells { C12 C13 }
```

The following example includes the cell named **C71** in the subdesign *MID*, so that only this cell is considered for clock gate replacement at that level of the hierarchy.

```
prompt> set_replace_clock_gates -include_cells MID/C71
```

The following example sets the edge type of clock input **clk** of the black-box instance **RAM_03** in the current design to positive edge triggered.

```
prompt> set_replace_clock_gates -rising_edge_clock RAM_03/clk
```

The following example achieves the same result by specifying the edge type directly on the port of the black-box design.

```
prompt> current_design RAM
prompt> set_replace_clock_gates -rising_edge_clock clk
prompt> current_design top_level
```

SEE ALSO

`replace_clock_gates(2)`
`report_clock_gating(2)`
`set_clock_gating_style(2)`

set_resistance

Sets the resistance value on nets.

SYNTAX

```
status set_resistance  
    value  
    [-min] [-max] net_list
```

Data Types

<i>value</i>	float
<i>net_list</i>	list

ARGUMENTS

value

Specifies the resistance value for nets in *net_list*. *value* must be expressed in unit system consistent with what the technology library used during optimization. For example, if the technology library specifies resistance values in ohms, *value* must also be expressed in ohms.

-min

Indicates that the resistance value is to be used for minimum delay analysis. If you do not specify a value for minimum delay analysis, the maximum value is used. If you do not specify a value for maximum delay analysis, the minimum value is ignored. (You cannot annotate only a minimum value.)

-max

Indicates that the resistance value is to be used for maximum delay analysis. If you do not specify **-min** or **-max**, the value is used for both minimum and maximum delay analysis.

net_list

Specifies a list of nets for which the specified resistance values are to be set.

DESCRIPTION

Sets the **ba_net_resistance** attribute, which enables the back-annotation of resistance values on nets in the current design. If the current design is hierarchical, it must have been linked with the **link** command. The specified *value* overrides the internally-estimated net resistance value.

You also can use the **set_resistance** command for nets at lower levels of the design hierarchy. These nets are specified as "BLOCK1/BLOCK2/NET_NAME."

To view resistance values, use the **report_net** command.

To remove a resistance value, use **remove_attribute**. To reset all back-annotated resistance values in a design, use the **reset_design** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example sets a resistance of 200 units to nets a and b.

```
prompt> set_resistance 200 {a b}
```

This example sets a resistance of 300 units on net U1/U2/NET3.

```
prompt> set_resistance 300 U1/U2/NET3
```

This example removes the back-annotated resistance on net U1/U2/NET3.

```
prompt> remove_attribute [get_nets U1/U2/NET3] ba_net_resistance
```

SEE ALSO

- `remove_attribute(2)`
- `report_net(2)`
- `report_timing(2)`
- `reset_design(2)`
- `set_drive(2)`
- `target_library(3)`

set_resource_allocation

Sets the `resource_allocation` attribute on the current design, specifying the type of resource allocation to be used by `compile`.

SYNTAX

integer **set_resource_allocation**
none | area_only | area_no_tree_balancing | constraint_driven

ARGUMENTS

none | **area_only** | **area_no_tree_balancing** | **constraint_driven**

Specifies the allocation type with which the **resource_allocation** attribute is to be set. The type values are as follows:

- **none** directs **compile** to use no resource sharing, so that each operation is implemented with separate circuitry.
- **area_only** directs **compile** to share operators without considering timing constraints while simultaneously balancing all arithmetic expression trees.
- **area_no_tree_balancing** directs **compile** to share operators without considering timing constraints and without balancing expression trees.
- **constraint_driven** (the default) directs **compile** to share resources so that timing constraints are met or not worsened by sharing.

DESCRIPTION

This command sets the **resource_allocation** attribute on the current design, specifying the type of resource allocation to be used by **compile**. The **synthetic_library** variable must be set for resource sharing to take affect. This command has no effect in **compile_ultra**.

If **resource_allocation** is not set, then the value of the **hlo_resource_allocation** environment variable is used. The default value for the variable is **constraint_driven**. By default, **compile** shares resources so that timing constraints are met or not worsened. Setting the **resource_allocation** attribute overrides the value of **hlo_resource_allocation** for the current design.

Use the **remove_attribute** command to undo this command.

EXAMPLES

The following example sets the **resource_allocation** attribute to **area_only** on the design named TEST:

```
prompt> current_design TEST  
prompt> set_resource_allocation area_only
```

The following example removes the **resource_allocation** attribute from the current design:

```
prompt> remove_attribute current_design resource_allocation
```

SEE ALSO

- `compile(2)`
- `get_attribute(2)`
- `remove_attribute(2)`
- `hlo_resource_allocation(3)`
- `synthetic_library(3)`

set_retention

Defines the UPF retention strategy for the power domains in the design.

SYNTAX

```
status set_retention
  retention_strategy
  -domain power_domain
  [-retention_power_net retention_power_net]
  [-retention_ground_net retention_ground_net]
  [-retention_supply_set retention_supply_set]
  [-retention_supply retention_supply_set]
  [-no_retention]
  [-elements objects]
  [-save_condition {boolean_function}]
  [-restore_condition {boolean_function}]
  [-retention_condition {boolean_function}]
  [-save_signal {save_signal save_sense}]
  [-restore_signal {restore_signal restore_sense}]
  [-update]
  [-use_retention_as_primary]
```

Data Types

<i>retention_strategy</i>	string
<i>power_domain</i>	string
<i>retention_power_net</i>	string
<i>retention_ground_net</i>	string
<i>retention_supply_set</i>	string
<i>objects</i>	list
<i>boolean_function</i>	string
<i>save_signal</i>	string
<i>save_sense</i>	high low
<i>restore_signal</i>	string
<i>restore_sense</i>	high low

ARGUMENTS

retention_strategy

Specifies the name of the UPF retention strategy. The name of the retention strategy must be unique within the specified power domain.

-domain *power_domain*

Specifies the name of the power domain to which this UPF retention strategy should be applied.

-retention_power_net *retention_power_net*

Specifies the retention power net for the retention cells that are in this UPF retention strategy.

-retention_ground_net *retention_ground_net*

Specifies the retention ground net for the retention cells that are in this UPF retention strategy.

-retention_supply_set *retention_supply_set*

Specifies the supply set whose power and ground function associations should to be used as the retention power and retention ground nets respectively. It is mutually exclusive with the **-retention_power_net** and the **-retention_ground_net** options. This option is supported with UPF 2.0 standard.

-retention_supply *retention_supply_set*

Specifies the same functionality as **-retention_supply_set**, except this option is supported with UPF 3.0 standard. Only one of **-retention_supply_set** or **-retention_supply** can be specified.

-no_retention

Specifies that the objects in the power domain do not have retention.

-elements *objects*

Specifies the objects to which this UPF retention strategy applies. The objects can be hierarchical cells, leaf cells, HDL blocks, register signals of a Verilog always block, or the name of a list of retention elements.

-save_condition {*boolean_function*}

This option controls the saving of the current register's value in the retention register's shadow latch.

-restore_condition {*boolean_function*}

The `restore_condition` controls the restoration of the value from the register's shadow latch to the register.

-retention_condition {*boolean_function*}

The `retention_condition` controls the preservation of the saved value in the shadow register.

-save_signal {*save_signal save_sense*}

Specifies the name and active logic state of an existing port, pin, or net that controls saving data into the retention elements before power-down. The logic state can be either **low** or **high**.

-restore_signal {*restore_signal restore_sense*}

Specifies the name and active logic state of an existing port, pin, or net that controls restoring data from the retention elements after power-up. The logic state can be either **low** or **high**.

-update

Updates the specification of an existing UPF retention strategy. This option can only update the element list of the strategy.

-use_retention_as_primary

When specified, tool must map the elements of this retention strategy to retention library cell whose output pin is related to retention supply.

In the absence of this option, tool will map the elements of this retention strategy to a retention library cell whose output is related to the primary of the domain.

DESCRIPTION

If the **-no_retention** option is not used, this command defines the UPF retention strategy on the specified power domain, to map the unmapped sequential cells to retention cells.

If the **-no_retention** is used, unmapped sequential cells are mapped to cells that do not have retention functionality. This option is useful for excluding descendant cells from a different retention strategy applied to a hierarchical cell.

If the **-elements** option is not specified, the retention strategy is applied to all the unmapped sequential cells in the power domain. If the **-elements** is specified, the UPF retention strategy is applied to all the unmapped sequential cells that are under the objects from **-elements**.

If neither the **-retention_supply_set** argument nor **-retention_power_net** or **-retention_ground_net** is specified, the retention power and ground nets are automatically set to the power and ground functions of the **default_retention** supply set handle of the power domain, for which the strategy is being defined.

This command specifies the save and restore signals and logical senses of those signals for a specified retention strategy. You can specify each control signal as a port, pin, or net. A port or pin has priority over an identically named net.

The retention signal need not exist in the logical hierarchy where the retention cells are be inserted. The synthesis or implementation tool can perform port-punching to make the connection. Port-punching means automatically creating a port to make a connection from one hierarchical level to the next.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to define a UPF retention strategy in the specified power domain PD1. Power domain PD1 is defined on instance shutdown_inst.

```
prompt> set_retention retention_1 \
  -domain PD1 \
  -retention_power_net PN1 B \
  -retention_ground_net GN1
```

The following example shows how to define a UPF retention strategy in the objects under the specified power domain:

```
prompt> set_retention retention_2 \
  -domain PD1 \
  -retention_power_net PN1 \
  -retention_ground_net GN1 \
  -elements shutdown_inst/mid_inst
```

The following example shows how to define a UPF retention strategy that excludes register R_reg from retention of its parent cell:

```
prompt> set_retention retention_3 \
  -domain PD1 \
  -no_retention \
  -elements shutdown_inst/mid_inst/R_reg
```


The following example shows how to define a UPF retention strategy using a supply set:

```
prompt> set_retention retention_2 -domain PD1 \
      -retention_supply_set ret_supply_set \
      -elements shutdown_inst/mid_inst/R_reg
```

The following example shows how a UPF retention strategy can be defined without using a supply set or supply nets.

```
prompt> set_retention retention_2 -domain PD1 \
      -elements shutdown_inst/mid_inst/R_reg
```

The following example shows how to define a UPF retention strategy with the save/restore/retention conditions:

```
prompt> set_retention retention_3 -domain PD1 \
      -elements shutdown_inst/mid_inst/R_reg \
      -save_condition {save_net&&restore_net} \
      -restore_condition {!restore_net} \
      -retention_condition {save_net||restore_net}
```

In the following example, suppose there is a RTL which results in registers with the names mygenblk[0].myblk.myreg_reg and mygenblk[1].myblk.myreg_reg:

```
4 generate for (i=0; i < 2; i = i + 1 ) begin : mygenblk
5   always @(posedge clk or negedge rst) begin : myblk
6     reg myreg;
7     if( !rst ) begin
8       qout = 0;
9       myreg = 0;
10    end
11  else begin
12    qout = myreg;
13    myreg = datain;
14  end
15 end
16 end endgenerate
```

The following example shows how to define a UPF retention strategy for registers using the HDL block name or register signal name:

```
prompt> set_retention retention_4 -domain PD1 \
      -elements mygenblk[*]
prompt> set_retention retention_5 -domain PD1 \
      -elements mygenblk[*].myblk
prompt> set_retention retention_6 -domain PD1 \
      -elements mygenblk[*].myblk.myreg
```

The following example shows how to specify save and restore signals in the **set_retention** command:

```
prompt> set_retention retention_4 -domain PD1 \
      -save_signal {sleep high} \
      -restore_signal {wake_up low} \
      -elements mygenblk[*]
```

The following example shows how to update a UPF retention strategy with new elements:

```
prompt> set_retention retention_1 -domain PD1 \
      -elements {mid_inst_1}
prompt> set_retention retention_1 -domain PD1 \
      -elements {mid_inst_2} -update
```

SEE ALSO

map_retention_cell(2)
set_retention_control(2)
report_retention_cell(2)
hdlin_enable_upf_compatible_naming(3)
hdlin_enable_hier_naming(3)
set_retention_elements(2)

set_retention_cell

Sets the specified library cells as retention cells.

SYNTAX

```
status set_retention_cell  
  cell_name  
  [-cell_type retention_type]  
  [-retention_pin {pin_name pin_type disable_value}]
```

Data Types

<i>cell_name</i>	string
<i>retention_type</i>	string
<i>pin_name</i>	string
<i>pin_type</i>	string
<i>disable_value</i>	Boolean

ARGUMENTS

cell_name

Specifies the name of the library cell to be defined as a retention cell.

-cell_type *retention_type*

Specifies the type of the retention cell.

-retention_pin {*pin_name pin_type disable_value*}

Defines the retention pin of the library cell. The valid values for the *pin_type* argument are restore, save, and save_restore. The value values for the *disable_value* argument are 0 and 1.

DESCRIPTION

The **set_retention_cell** command sets the specified library cells as retention cells. You can also specify the retention pin and retention cell type for the retention cell.

EXAMPLES

The following example sets all library cells whose name matches *DRFF* as DRFF retention cells. The retention pin, RETN, is a save-restore pin and is disabled by a logic 1 value.

```
prompt> set_retention_cell *DRFF* -cell_type DRFF \  
-retention_pin {RETN save_restore 1}
```

SEE ALSO

report_lib(2)
check_library(2)

set_retention_control

Defines the UPF retention control signals for a retention strategy.

SYNTAX

```
status set_retention_control
    retention_strategy
    -domain power_domain
    -save_signal {save_signal save_sense}
    -restore_signal {restore_signal restore_sense}
    [-assert_r_mutex {net_name sense}]
    [-assert_s_mutex {net_name sense}]
    [-assert_rs_mutex {net_name sense}]
```

Data Types

<i>retention_strategy</i>	string
<i>power_domain</i>	string
<i>save_signal</i>	string
<i>save_sense</i>	high low
<i>restore_signal</i>	string
<i>restore_sense</i>	high low
<i>net_name</i>	string
<i>sense</i>	high low

ARGUMENTS

retention_strategy

Specifies the UPF retention strategy name. The retention strategy should already be defined using the **set_retention** command.

-domain *power_domain*

Specifies the power domain where this UPF retention strategy is applied.

-save_signal {*save_signal* *save_sense*}

Specifies an existing port, pin, or net in the design used to save data into the shadow register before power-down; and the logic state of the signal, either **low** or **high**, that causes this action to be taken.

-restore_signal {*restore_signal* *restore_sense*}

Specifies an existing port, pin, or net in the design used to restore data from the shadow register before power-up; and the logic state of the signal, either **low** or **high**, that causes this action to be taken.

-assert_r_mutex {*net_name* *sense*}

Creates an assertion, which the verification tools can trigger, when the specified RTL signal is active simultaneously with the restore signal.

-assert_s_mutex {*net_name sense*}

Creates an assertion, which the verification tools can trigger, when the specified RTL signal is active simultaneously with the save signal.

-assert_rs_mutex {*net_name sense*}

Creates an assertion, which the verification tools can trigger, when the specified RTL signal is active simultaneously with the save or restore signals. If the signal name and value are not specified, it indicates the save and restore signals are mutually exclusive.

DESCRIPTION

This command specifies the save and restore signals and logical senses of those signals for a specified retention strategy. You can specify each control signal as a port, pin, or net. A port or pin has priority over an identically named net.

The retention signal need not exist in the logical hierarchy where the retention cells are be inserted; the synthesis or implementation tool can perform port-punching as needed to make the connection. Port-punching means automatically creating a port to make a connection from one hierarchical level to the next. These punched ports are not considered for isolation or level-shifting, even though after the port creation, these ports reside within the coverage of an isolation or level-shifter strategy. Existing ports, even if they reside on an always-on path, are isolated and level-shifted according to the applicable isolation and level-shifter strategy.

The **-assert_r_mutex**, **-assert_s_mutex**, and **-assert_rs_mutex** options create assertions, which can be used as triggers by the verification and simulation tools. They are read by synthesis and implementation tools, but they are not used by these tools and are not written out by the **save_upf** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to define the retention control signals for the retention strategy `retention_1` of the power domain `PD1`.

```
prompt> set_retention_control retention_1 \
-domain PD1 \
-save_signal {sleep high} \
-restore_signal {wake_up low}
```

When the signal called `sleep` goes high, it saves the data in the retention register for entry into the power-down state. Upon exit from the power-down state, when the signal called `wake_up` goes low, it restores the data to the retention register. Each signal name can be port, pin, or net.

The following example shows how to define the assertion options for the retention strategy `retention_2` of the power domain `PD1`.

```
prompt> set_retention_control retention_2 \
-domain PD1 \
-save_signal {sleep high} \
```

```
-restore_signal {wake_up low} \  
-assert_r_mutex {wake_up high} \  
-assert_s_mutex {sleep low} \  
-assert_rs_mutex {wake_up low}
```

The following example shows how to define the assertion options when the save signal and restore signal are mutually exclusive.

```
prompt> set_retention_control retention_3 \  
-domain PD1 \  
-save_signal {sleep high} \  
-restore_signal {wake_up low} \  
-assert_rs_mutex {}
```

SEE ALSO

map_retention_cell(2)
set_retention(2)

set_retention_control_pins

Converts the retention register library cell attributes in the old library format to the ones that can be used in \$retain flow. The \$retain flow requires retention register library cells to have new retention cell attributes, which is different from the original power gating flow.

SYNTAX

```
status set_retention_control_pins
  [-type style]
  [-power_pin_index power_pin
   | -library_pin library_pin_name]
  [-is_save_pin
   | -is_restore_pin
   | -is_save_restore_pin]
  lib_or_lib_cell_list
```

Data Types

<i>style</i>	string
<i>power_pin</i>	string
<i>library_pin_name</i>	string
<i>lib_or_lib_cell_list</i>	string

ARGUMENTS

-type *style*

Specifies the retention register style.

-power_pin_index *power_pin*

Specifies the power pin index number of the old retention register library pin.

-library_pin *library_pin_name*

Specifies the name of the library pin of the old retention register cell.

-is_save_pin

Assigns the specified power pin index or library pin as a SAVE pin.

-is_restore_pin

Assigns the specified power pin index or library pin as a RESTORE pin.

-is_save_restore_pin

Assigns the specified power pin index or library pin as a SAVE_RESTORE pin.

lib_or_lib_cell_list

Specifies the retention register cell library or a list of library cells to which the new attributes are applied.

DESCRIPTION

The \$retain flow requires retention register library cells to have new retention cell attributes, which is different from the original power gating flow. This command converts the retention register library cell attributes in the old library format to the ones that can be used in \$retain flow.

EXAMPLES

The command should be specified before running the **compile** command. The following example sets the new attributes to the list of specified library cells:

```
prompt>set_retain_cell_CLK_LOW_lat \  
[get_lib_cell GS60_W_125_1.08_CORE_RET_SNPB.db/RTCLLAH*]  
  
prompt>set_retain_cell_CLK_LOW_reg \  
[get_lib_cell GS60_W_125_1.08_CORE_NSFLOP_RET_SNPB.db/RTCLDFF*]  
  
prompt>set_retention_control_pins -type CLK_LOW \  
-power_pin_index 1 -is_save_pin $retain_cell_CLK_LOW_lat  
  
prompt>set_retention_control_pins -type CLK_LOW -power_pin_index 2 \  
-is_restore_pin $retain_cell_CLK_LOW_lat  
  
prompt>set_retention_control_pins -type CLK_LOW -power_pin_index 1 \  
-is_save_pin $retain_cell_CLK_LOW_reg  
  
prompt>set_retention_control_pins -type CLK_LOW -power_pin_index 2 \  
-is_restore_pin $retain_cell_CLK_LOW_reg
```

SEE ALSO

set_retention_control_pins(2)

set_retention_elements

Creates a named list of elements that can be used in the `set_retention` command.

SYNTAX

```
status set_retention_elements  
  retention_list_name  
  -elements objects
```

Data Types

<i>retention_list_name</i>	string
<i>objects</i>	list

ARGUMENTS

retention_list_name

Specifies name of the retention element list. The name must be unique within the current scope.

-elements *objects*

Specifies the objects that belong to the retention element list. The objects can be hierarchical cells, leaf cells, HDL blocks, or register signals of a Verilog always block. The objects list can include dot ".".

DESCRIPTION

Creates a named list of elements at the current scope. Elements must adhere to the following rules:

- It is an error if an element belonging to the retention element list is not retained when any element in the same list is retained.
- It is an error if an element belonging to the retention element list is not retained when any element in the same power domain extent is retained.
- It is an error to specify **no_retention** constraints on an element belonging to the retention element list.

If dot "." is in the element list, then all the elements in the current scope should adhere to the previously mentioned rules.

NOTE: Specifying **set_retention_elements** does not mandate retaining the elements.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to define a retention element list.

```
prompt> set_retention_elements RET_LIST1 \  
-elements shutdown_inst/mid_inst
```

SEE ALSO

map_retention_cell(2)
set_retention(2)

set_route_zrt_common_options

Sets the options common to all phases of Zroute routing. Options set are saved with the design.

SYNTAX

```
status set_route_zrt_common_options
  [-reroute_clock_shapes true | false]
  [-reroute_user_shapes true | false]
  [-plan_group_aware off | all_routing | top_level_routing_only]
  [-reshield_modified_nets off | reshield | unshield]
  [-child_process_net_threshold inf]
  [-verbose_level inf]
  [-clock_topology normal | comb]
  [-comb_distance inf]
  [-pg_shield_distance_threshold distance]
  [-connect_floating_shapes true | false]
  [-allow_pg_as_shield true | false]
  [-rc_driven_setup_effort_level off | low | medium | high]
  [-read_user_metal_blockage_layer true | false]
  [-wide_macro_pin_as_fat_wire true | false]
  [-route_top_boundary_mode stay_half_min_space_inside | stay_inside | ignore]
  [-standard_cell_blockage_as_thin true | false]
  [-freeze_layer_by_layer_name {{layer true | false}...}]
  [-freeze_via_to_frozen_layer_by_layer_name {{layer true | false}...}]
  [-forbid_new_metal_by_layer_name {{layer true | false}...}]
  [-global_max_layer_mode soft | allow_pin_connection | hard]
  [-global_min_layer_mode soft | allow_pin_connection | hard]
  [-ignore_var_spacing_to_blockage true | false]
  [-ignore_var_spacing_to_pg true | false]
  [-ignore_var_spacing_to_shield true | false]
  [-min_edge_offset_for_macro_pin_connection_by_layer_name {{layer offset}...}]
  [-net_max_layer_mode soft | allow_pin_connection | hard]
  [-net_min_layer_mode soft | allow_pin_connection | hard]
  [-net_max_layer_mode_soft_cost low | medium | high]
  [-net_min_layer_mode_soft_cost low | medium | high]
  [-extra_preferred_direction_wire_cost_multiplier_by_layer_name {{layer multiplier}...}]
  [-extra_nonpreferred_direction_wire_cost_multiplier_by_layer_name {{layer multiplier}...}]
  [-extra_via_cost_multiplier_by_layer_name {{layer multiplier}...}]
  [-extra_via_off_grid_cost_multiplier_by_layer_name {{layer multiplier}...}]
  [-number_of_vias_over_global_max_layer inf]
  [-number_of_vias_under_global_min_layer inf]
  [-number_of_vias_over_net_max_layer inf]
  [-number_of_vias_under_net_min_layer inf]
  [-via_array_mode off | swap | rotate | all]
  [-post_detail_route_redundant_via_insertion off | low | medium | high]
  [-post_incremental_detail_route_fix_soft_violations true | false]
  [-concurrent_redundant_via_mode off | reserve_space | insert_at_high_cost]
```

```

[-concurrent_redundant_via_effort_level low | medium | high]]
[-eco_route_concurrent_redundant_via_mode off | reserve_space
[-eco_route_concurrent_redundant_via_effort_level low | medium | high]]
[-rotate_default_vias true | false]
[-route_soft_rule_effort_level off | min | low | medium | high]
[-post_detail_route_fix_soft_violations true | false]
[-post_eco_route_fix_soft_violations true | false]
[-post_group_route_fix_soft_violations true | false]
[-routing_rule_effort_level {{rule {layer {effort_list}...}...}}]
[-soft_rule_weight_to_effort_level_map {{weight_level effort_level}...}]
[-enforce_voltage_areas off | strict | relaxed]
[-voltage_area_weight {{voltage_area_name weight}...}]
[-single_connection_to_pins connection_mode]
[-mark_clock_nets_minor_change true | false]
[-track_auto_fill true | false]
[-tie_off_mode all | rail_only]
[-connect_within_pins_by_layer_name {{layer mode}...}]
[-number_of_secondary_pg_pin_connections int]
[-separate_tie_off_from_secondary_pg true | false]
[-default true | false]
[-shielding_nets pg_net_names]
[-report_local_double_pattern_odd_cycles true | false]
[-wire_on_grid_by_layer_name {{layer true | false}...}]
[-via_on_grid_by_layer_name {{layer true | false}...}]
[-min_shield_length_by_layer_name {{layer length}...}]

```

Data Types

<i>int</i>	integer
<i>layer</i>	string
<i>offset</i>	float
<i>multiplier</i>	float
<i>rule</i>	string
<i>effort_list</i>	list of strings
<i>voltage_area_name</i>	string
<i>weight</i>	string
<i>connection_mode</i>	string
<i>mode</i>	string
<i>pg_net_names</i>	list of strings
<i>length</i>	float

ARGUMENTS

-reroute_clock_shapes true | false

Specifies whether the router can reroute fixed clock wires and vias.

The default is **false**.

-reroute_user_shapes true | false

Specifies whether the router can reroute user-created wires and vias.

The default is **false**.

-plan_group_aware off | all_routing | top_level_routing_only

Specifies whether the plan-group constraints are obeyed during global routing. This parameter does not affect track assignment or detail routing.

The definition for each mode is

- **off** (the default)
The global router ignores plan groups and routes the design flat.
- **all_routing**
The global router routes all nets and preserves the hierarchy and pin constraints for the pin-cutting flow.
- **top_level_routing_only**
The global router preserves the hierarchy and pin constraints, but it ignores intraplan group nets and routes only those nets that have top-level connections. This mode saves CPU time, but might be less accurate because it ignores the congestion due to intraplan group nets.

When you specify a mode other than **off**, the global router gives priority to clock net routing. During floorplanning, you might want to route the clocks flat to generate clock pins on the module boundaries. You should buffer these clocks later in the flow. The clock nets are routed before any other nets so that their pin placement has priority over normal signal pins. Timing and cross talk optimization are not supported when plan group aware routing is active. The "comb" topology for clocks is also unsupported in plan group aware routing.

-reshield_modified_nets off | reshield | unshield

Specifies whether to automatically unshield or reshield modified shielded nets during routing.

The definition for each mode is

- **off** (the default)
The router does not unshield or reshield modified shielded nets.
- **reshield**
The router unshields modified shielded nets, automatically reshields them, and then reconnects the floating shielding. If the router cannot reconnect the floating shielding, it removes it. Shielded nets might have been modified by routing or outside of routing by manual editing, route optimization, and other commands.
- **unshield**
The router deletes existing shielding that is associated with modified shielded nets and reconnects the floating shielding if necessary. The nets might be reshielded at a later stage.

-child_process_net_threshold *int*

Specifies the minimum number of nets to trigger a separate router process.

The default is -1, which means that the tool decides when to trigger a separate router process based on the number of nets in the design and the peak memory of the parent process when the routing command is run.

-verbose_level *int*

Sets the verbosity level for the routing log file.

The value range is from 0 to 2.
The default is 0.

-clock_topology normal | comb

Specifies the clock routing topology. Valid values are

- **normal** (default)

Use normal routing for clock nets.

- **comb**

Use comb routing for clock nets with a mesh or trunk.

-comb_distance *int*

Specifies the number of global routing cells within which to connect clock pins to the clock mesh.

The range is from 0 to 10.

The default is 2.

-pg_shield_distance_threshold *distance*

Specifies the distance threshold in microns for the router to consider the existing power and ground structure as shielding when calculating the shielding ratio. When a PG wire is within the specified distance, it is counted as a shielding wire.

Note that this option does not change the routing behavior. It affects only the shielding ratio report generated by the **create_zrt_shield** and **report_zrt_shield** commands.

The default distance is 0, which means that the router determines the distance threshold.

-connect_floating_shapes *true | false*

Specifies whether the router should connect just the pins or the pins and all floating shapes.

When **false** (the default), the router connects just the pins. The router might use the floating shapes; however, it does not have to connect the floating shapes.

When **true**, the router connects the pins and all floating shapes.

-allow_pg_as_shield *true | false*

Specifies whether the router should treat power and ground nets as shields.

When **true** (the default), the router treats power and ground nets as shields and does not reserve additional space around the nets to be shielded.

If set to **false**, the router tries to reserve space around the nets to be shielded if they are close to a power or ground net.

Note that this option changes the routing behavior.

-rc_driven_setup_effort_level *off | low | medium | high*

Enables the router to automatically bias layer preference based on RC values for better setup timing.

The default setting is **medium**.

This option takes effect only if timing-driven routing is enabled.

-read_user_metal_blockage_layer *true | false*

Specifies how the router treats shapes on metal blockage layers.

When **false** (the default), the shapes are ignored.

When **true**, the shapes are honored.

-wide_macro_pin_as_fat_wire *true | false*

Specifies whether a wide macro pin or pad pin should have an implicit depth the same as its width.

The default value is **false**.

-route_top_boundary_mode stay_half_min_space_inside | stay_inside | ignore

Controls the routing behavior near the top boundary.

The definition for each mode is

- **stay_half_min_space_inside**
The routing must be inside the boundary by half of the minimum spacing.
- **stay_inside** (the default)
The routing must be inside or abut the boundary.
- **ignore**
Wires can extend outside the boundary.

-standard_cell_blockage_as_thin true | false

Specifies whether to treat wide blockages in standard cells as thin wires.

The default is **false**.

-freeze_layer_by_layer_name {{layer true | false}...}

Controls whether routing layers are frozen to prevent them from changing during routing. You must specify the routing layers by using the layer names in the technology file.

This option can be set only after the cell is open.

When **false** (the default), the layer is not frozen.

When **true**, the layer is frozen.

-freeze_via_to_frozen_layer_by_layer_name {{layer true | false}...}

Controls the treatment of vias that touch the specified frozen layers on one side. You must specify the routing layers by using the layer names in the technology file. The specified layers should be frozen layers.

By default, all the frozen layers are set as false, which indicates that the vias whose surroundings on a frozen layer are enclosed by other metal shapes can be removed or rerouted as long as they do not make a change on the frozen metal layer. Special vias such as H-shape vias are not removed or rerouted even though they do not change the frozen metal layer.

When you set this option to **true** on one or more frozen layers, the vias adjacent to these frozen layers are frozen and cannot be changed. Note that no change is allowed on a cut layer adjacent to two frozen metal layers.

-forbid_new_metal_by_layer_name {{layer true | false}...}

Controls whether ECO routing (the **route_zrt_eco** command) can remove, but not add, metal on the specified layers. You must specify the routing layers by using the layer names in the technology file.

Note that for all other routing commands, this option acts the same as the **-freeze_layer_by_layer_name** option and controls whether routing cannot add or remove metal on the specified layers.

When **false** (the default), the router can add or remove metal from the layer.

When **true**, the ECO router can remove, but not add, metal from the layer. The ECO router can remove metal if it is dangling or is not fully contained within fixed user shapes, such as pins. All other routing phases cannot add or remove metal from the layer. If the router adds metal on the layer, and that metal is not fully contained within a fixed user shape, the router reports a "Forbidden metal" DRC violation.

You would use this option if you want to essentially freeze a layer when running core commands, such as **route_opt**, but give the router the flexibility to remove some shapes during ECO routing. If a layer is frozen with the **-freeze_layer_by_layer_name** option, the router cannot touch any shapes, which can result in dangling wires and unfixable shorts if the placement changes

during ECO. The **-forbid_new_metal_by_layer_name** option gives the router the flexibility to remove such shapes while otherwise treating the layer as frozen.

If you specify both the **-freeze_layer_by_layer_name** and **-forbid_new_metal_by_layer_name** options, the stricter **-freeze_layer_by_layer_name** option takes precedence.

This option can be set only after the cell is open.

-global_max_layer_mode soft | allow_pin_connection | hard

Controls the application of the maximum routing layer constraint set for the design. This global constraint is set by using the **set_ignored_layers** command.

The definition for each mode is

- **soft** (the default)
Routing above the maximum layer is discouraged, but not disallowed. DRC violations are not flagged on any metal above the maximum layer.
- **allow_pin_connection**
Routing is allowed above the maximum layer only for pin connections. DRC violations are flagged on metal above the maximum layer if the metal is not along a path that connects to a pin. DRC violations are flagged on metal above the maximum layer if the metal is along a path that connects to a pin and the path is long. A path is considered long by the detail router if its length exceeds approximately 10 average track pitches. A path is considered long by the **verify_zrt_route** command if its length exceeds approximately 15 average track pitches.
- **hard**
No routing can occur above the maximum layer. DRC violations are flagged on metal above the maximum layer.

Note that the **-number_of_vias_over_global_max_layer** option can relax the maximum layer constraint, but only for vias.

-global_min_layer_mode soft | allow_pin_connection | hard

Controls the application of the minimum routing layer constraint for the design. This global constraint is set by using the **set_ignored_layers** command.

The definition for each mode is

- **soft** (the default)
Routing below the minimum layer is discouraged, but not disallowed. DRC violations are not flagged on any metal below the minimum layer.
- **allow_pin_connection**
Routing is allowed below the minimum layer only for pin connections. DRC violations are flagged on metal below the minimum layer if the metal is not along a path that connects to a pin. DRC violations are flagged on metal below the minimum layer if the metal is along a path that connects to a pin and the path is long. A path is considered long by the detail router if its length exceeds approximately 10 average track pitches. A path is considered long by the **verify_zrt_route** command if its length exceeds approximately 15 average track pitches.
- **hard**
No routing can occur below the minimum layer. DRC violations are flagged on metal below the minimum layer.

Note that the **-number_of_vias_under_global_min_layer** option can relax the minimum layer constraint, but only for vias.

-ignore_var_spacing_to_blockage true | false

Controls whether the spacing defined in a nondefault routing rule is ignored against blockages.

The default is **false**.

-ignore_var_spacing_to_pg true | false

Controls whether the spacing defined in a nondefault routing rule is ignored against all wires on power and ground nets.

The default is **false**.

-ignore_var_spacing_to_shield true | false

Controls whether the spacing defined in a nondefault routing rule is ignored against shield wires on power and ground nets.

The default is **true**.

This option has no effect if the **-ignore_var_spacing_to_pg** option is **true**.

-min_edge_offset_for_macro_pin_connection_by_layer_name {{layer offset}...}

Specifies the minimum edge offset for macro pin connections for one or more metal layers. The minimum edge for macro pin connection rule enforces exactly aligned connections between metal wires and macro pins where they connect. If the macro pin and wire have different widths, the short edges resulting at the connection point must have at least the specified length.

The settings specified in this option override the **minEdgeOffsetForMacroPinConnection** values specified in the Layer section of the technology file for the specified layers. The technology file values are used for any layers not specified in this option.

-net_max_layer_mode soft | allow_pin_connection | hard

Defines the default mode for the net-based maximum routing layer constraints set by the **set_net_routing_layer_constraints** command. This setting applies only to nets whose maximum routing layer mode is not set by the **set_net_routing_layer_constraints -max_layer_mode** command option.

The definition for each mode is

- **soft**
Routing above the maximum layer is discouraged, but not disallowed. DRC violations are not flagged on any metal above the maximum layer.
- **allow_pin_connection**
Routing is allowed above the maximum layer only for pin connections. DRC violations are flagged on metal above the maximum layer if the metal is not along a path that connects to a pin. DRC violations are flagged on metal above the maximum layer if the metal is along a path that connects to a pin and the path is long. A path is considered long by the detail router if its length exceeds approximately 10 average track pitches. A path is considered long by the **verify_zrt_route** command if its length exceeds approximately 15 average track pitches.
- **hard** (the default)
No routing can occur above the maximum layer. DRC violations are flagged on metal above the maximum layer.

Note that the **-number_of_vias_over_net_max_layer** option can relax the maximum layer constraint, but only for vias.

-net_min_layer_mode soft | allow_pin_connection | hard

Defines the default mode for the net-based minimum routing layer constraints set by the **set_net_routing_layer_constraints** command. This setting applies only to nets whose minimum routing layer mode is not set by the **set_net_routing_layer_constraints -min_layer_mode** command option.

The definition for each mode is

- **soft** (the default)
Routing below the minimum layer is discouraged, but not disallowed. DRC violations are not flagged on any metal below the minimum layer.
- **allow_pin_connection**
Routing is allowed below the minimum layer only for pin connections. DRC violations are flagged on metal below the minimum layer if the metal is not along a path that connects to a pin. DRC violations are flagged on metal below the minimum layer if the metal is along a path that connects to a pin and the path is long. A path is considered long by the detail

router if its length exceeds approximately 10 average track pitches. A path is considered long by the **verify_zrt_route** command if its length exceeds approximately 15 average track pitches.

- **hard**

No routing can occur below the minimum layer. DRC violations are flagged on metal below the minimum layer.

Note that the **-number_of_vias_under_net_min_layer** option can relax the minimum layer constraint, but only for vias.

-net_max_layer_mode_soft_cost *low | medium | high*

Defines the default cost for routing above the net-based maximum routing layer when using the soft maximum layer mode. This setting applies only to nets whose soft mode maximum routing layer cost is not set by the **set_net_routing_layer_constraints - max_layer_mode_soft_cost** command option.

The default is **medium**.

-net_min_layer_mode_soft_cost *low | medium | high*

Defines the default cost for routing below the net-based minimum routing layer when using the soft minimum layer mode. This setting applies only to nets whose soft mode minimum routing layer cost is not set by the **set_net_routing_layer_constraints - min_layer_mode_soft_cost** command option.

The default is **medium**.

-extra_preferred_direction_wire_cost_multiplier_by_layer_name *{{layer multiplier}...}*

Specifies the cost multiplier for routing in the preferred direction for the specified metal layers. You must specify the metal layers by using the layer names in the technology file.

The multiplier value must be between 0.0 and 20.0; it affects the routing cost in the preferred direction as follows:

$$\text{cost} = \text{baseCost} * (1 + \text{multiplier})$$

The default is 0.0.

You can set this option only after opening the top-level design.

-extra_nonpreferred_direction_wire_cost_multiplier_by_layer_name *{{layer multiplier}...}*

Specifies the cost multiplier for routing in the nonpreferred direction for the specified metal layers. You must specify the metal layers by using the layer names in the technology file.

The multiplier value must be between 0.0 and 20.0; it affects the routing cost in the nonpreferred direction as follows:

$$\text{cost} = \text{baseCost} * (1 + \text{multiplier})$$

The default is 0.0.

Note that specifying large values can prevent nonpreferred direction routing. If you require nonpreferred direction routing, you should use smaller values.

You can set this option only after opening the top-level design.

-extra_via_cost_multiplier_by_layer_name *{{layer multiplier}...}*

Specifies the cost multiplier for vias on the specified via layers. You must specify the via layers by using the layer names in the technology file.

The multiplier value must be between 0.0 and 20.0; it affects the via cost as follows:

$$\text{viaCost} = \text{baseViaCost} * (1 + \text{multiplier})$$

The default is 0.0.

You can set this option only after opening the top-level design.

-extra_via_off_grid_cost_multiplier_by_layer_name {{*layer multiplier*}...}

Specifies the cost multiplier for vias that are off grid with respect to the specified metal layers. You must specify the metal layers by using the layer names in the technology file.

The multiplier value must be between 0.0 and 20.0; it affects the via cost on the via layers adjacent to the specified metal layers as follows:

$$\text{viaCost} = \text{baseViaCost} * (1 + \text{multiplier})$$

The default is 0.0.

You can set this option only after opening the top-level design.

-number_of_vias_over_global_max_layer *int*

Allows the router to use stacked vias to access pins or fixed routes above the global maximum layer. If the global maximum layer mode is set to **soft**, this option is ignored, because in that case access above the maximum layer is allowed.

The value range is between 0 and 15. This value specifies the number of via layers that may be legally traversed to access a pin. For example, if the max layer is M6 and a value of 2 is given, a violation will not be flagged on vias on M6->M7 and M7->M8 if these vias are needed to access pins on layers above the max layer.

The default is 1.

-number_of_vias_under_global_min_layer *int*

Allows the router to use stacked vias to access pins or fixed routes below the global minimum layer. If the global minimum layer mode is set to **soft**, this option is ignored, because in that case access below the minimum layer is allowed.

The value range is between 0 and 15. This value specifies the number of via layers that may be legally traversed to access a pin. For example, if the min layer is M2 and a value of 1 is given, a violation will not be flagged on any vias on M1->M2 if these vias are needed to access pins on M1.

The default is 1.

-number_of_vias_over_net_max_layer *int*

Allows the router to use stacked vias to access pins or fixed routes above the net-specific maximum layer. If the net-specific maximum layer mode is set to **soft**, this option is ignored, because in that case access above the maximum layer is allowed.

The value range is between 0 and 15. This value specifies the number of via layers that may be legally traversed to access a pin. For example, if the max layer is M6 and a value of 2 is given, a violation will not be flagged on vias on M6->M7 and M7->M8 if these vias are needed to access pins on layers above the max layer.

The default is 1.

-number_of_vias_under_net_min_layer *int*

Allows the router to use stacked vias to access pins or fixed routes below the net-specific minimum layer. If the net-specific minimum layer mode is set to **soft**, this option is ignored, because in that case access below the minimum layer is allowed.

The value range is between 0 and 15. For example, if the min layer is M2 and a value of 1 is given, a violation will not be flagged on any vias on M1->M2 if these vias are needed to access pins on M1.

The default is 1.

-via_array_mode *off* | *swap* | *rotate* | *all*

Specifies the types of rotated via arrays that can be used during signal routing and redundant via insertion when using the default

routing rules.

The definition for each mode is

- **off**
Do not rotate or swap via arrays.
- **swap**
Use 1 x N and N x 1 via arrays as rotated equivalent vias.
- **rotate**
Rotate line via arrays instead of swapping row and column numbers.
- **all** (the default)
Use all four possible via arrays.

-post_detail_route_redundant_via_insertion off | low | medium | high

Enables automatic redundant via insertion after each detail routing step.

The default setting is **off**.

If the value is set to **low**, **medium**, or **high**, the tool performs redundant via insertion after each detail routing change, including initial detail routing, ECO routing, and incremental routing. Enabling this option keeps the redundant vias in the design up-to-date with routing changes. The **low**, **medium**, or **high** setting specifies the effort level used for redundant via insertion.

-post_incremental_detail_route_fix_soft_violations true | false

Enables automatic optimization of soft rules that are fixable postroute, after the incremental routing step.

The default setting is **false**.

-concurrent_redundant_via_mode off | reserve_space | insert_at_high_cost

Specifies the concurrent redundant via insertion mode for initial routing.

The valid modes are

- **off** (the default)
Concurrent redundant via insertion is turned off.
- **reserve_space**
Concurrent redundant via insertion reserves space for redundant vias but does not instantiate them. The space reservation is done by generating internal soft rules for redundant vias and performing soft-rule-based optimization. The actual instantiation is done later by explicitly calling the **insert_zrt_redundant_vias** command or by setting the **-post_detail_route_redundant_via_insertion** common Zroute option to a value other than **off**.
- **insert_at_high_cost**
Concurrent redundant via insertion creates hard design rules for redundant vias and tries to insert them. Use this mode if only you need a nearly 100 percent conversion rate. It can lead to very long runtimes if the design is congested or the technology parameters are not friendly to redundant via insertion. This mode should be used with caution.

-concurrent_redundant_via_effort_level low | medium | high

Specifies the effort level for the concurrent redundant via insertion flow.

The default effort level is **low**. The higher effort levels result in a better redundant via conversion rate at the expense of runtime.

This option takes effect only if the **-concurrent_redundant_via_mode** option is set to a value other than **off**.

-eco_route_concurrent_redundant_via_mode off | reserve_space

Specifies the concurrent redundant via insertion mode for ECO routing.

The definition for each mode is

- **off** (the default)
Concurrent redundant via insertion is turned off.
- **reserve_space**
Concurrent redundant via insertion reserves space for redundant vias in ECO routing but does not instantiate them. The space reservation is done by generating internal soft rules for redundant vias and performing soft-rule-based optimization. The actual instantiation is done later by explicitly calling the **insert_zrt_redundant_vias** command or by setting the **-post_detail_route_redundant_via_insertion** common Zroute option to a value other than **off**.

-eco_route_concurrent_redundant_via_effort_level low | medium | high

Specifies the effort level for the concurrent redundant via insertion flow in ECO routing.

The default effort level is **low**. The higher effort levels result in a better redundant via conversion rate at the expense of runtime.

This option takes effect only if the **-eco_concurrent_redundant_via_mode** option is set to a value other than **off**.

-rotate_default_vias true | false

Specifies whether the router can rotate default vias.

The default is **true**.

-route_soft_rule_effort_level off | min | low | medium | high

Specifies the effort level for rip up and reroute to fix soft rule design rule violations.

The definition for each effort level is

- **off**
Does not perform rip up and reroute passes to resolve soft rule design rule violations.
- **min**
Uses the smallest number of rip up and reroute passes to resolve soft rule design rule violations.
- **low**
Uses a small number of rip up and reroute passes to resolve soft rule design rule violations.
- **medium** (the default)
Uses a medium number of rip up and reroute passes to resolve soft rule design rule violations.
- **high**
Treats soft rule design rule violations the same as regular design rule violations during rip up and reroute.

-post_detail_route_fix_soft_violations true | false

Enables automatic optimization of soft rules that are fixable postroute, after the detail routing step.

The default setting is **false**.

-post_eco_route_fix_soft_violations true | false

Enables automatic optimization of soft rules that are fixable postroute, after the ECO routing step.

The default setting is **false**.

-post_group_route_fix_soft_violations true | false

Enables automatic optimization of soft rules that are fixable postroute, after the group routing step.

The default setting is **false**.

-routing_rule_effort_level {{rule {layer {effort_list}}...}...}

Specifies detailed, layer-by-layer effort levels on the various spacings of soft nondefault routing rules. For each nondefault routing rule, you can specify the effort for each spacing value defined for a layer. The rule name corresponds to a previously defined nondefault routing rule. For each layer defined in the **-spacings** option of the **define_routing_rule** command, you specify the effort level for each of the spacings defined for that layer.

The definition for each effort level is

- **low**
Uses a small number of rip up and reroute passes to resolve soft rule design rule violations.
- **med**
Uses a medium number of rip up and reroute passes to resolve soft rule design rule violations.
- **high**
Treats soft rule design rule violations the same as regular design rule violations during rip up and reroute.

For example, suppose you have defined the following two soft routing rules:

```
define_routing_rule soft_rule_1 \
  -spacings {M1 {0.12 0.24 }
            M2 {0.14 0.28 }
            M3 {0.14 0.28 }
            M4 {0.14 0.28 }
            M5 {0.14 0.28 }} \
  -spacing_weights {M1 {1 0.9 }
                   M2 {1 0.9 }
                   M3 {1 0.9 }
                   M4 {1 0.9 }
                   M5 {1 0.9 }}

define_routing_rule soft_rule_2 \
  -spacings {M1 {0.24 0.36}
            M2 {0.28 0.42}
            M3 {0.28 0.42}
            M4 {0.28 0.42}
            M5 {0.28 0.42}} \
  -spacing_weights {M1 {0.9 0.8}
                   M2 {0.9 0.8}
                   M3 {0.9 0.8}
                   M4 {0.9 0.8}
                   M5 {0.9 0.8}}
```

You would specify the detailed effort levels for these rules as shown in the following example:

```
set_route_zrt_common_options \
  -routing_rule_effort_level \
    { {soft_rule_1 { {M1 {high med}}
                     {M2 {high med}}
                     {M3 {high med}}
                     {M4 {high med}}
                     {M5 {high med}}
                     }
      }
    {soft_rule_2 { {M1 {high med}}
                   {M2 {low high}}
                   }
      }
  }
```

```

        {M3 {med high}}
        {M4 {med high}}
        {M5 {high high}}
    }
}

```

-soft_rule_weight_to_effort_level_map {{*weight_level effort_level*}...}

Specifies a global mapping from soft weight level to soft effort level.

The weight level must be one of the following values: low, medium, or high.

The effort level must be one of the following values: off, low, medium, or high.

The definition for each effort level is

- **off** (the default)
Does not perform rip up and reroute passes to resolve soft rule design rule violations.
- **low**
Uses a small number of rip up and reroute passes to resolve soft rule design rule violations.
- **medium**
Uses a medium number of rip up and reroute passes to resolve soft rule design rule violations.
- **high**
Treats soft rule design rule violations the same as regular design rule violations during rip up and reroute.

This mapping is applied whenever soft rules are defined using the **-spacing_weight_levels** option of the **define_routing_rule** command.

Each weight level should appear only once in the map.

The following mappings are not allowed:

- Low weight with medium effort
- Low weight with high effort

For example, suppose you have defined the following mapping:

```

set_route_zrt_common_options \
  -soft_rule_weight_to_effort_map \
    { {low low}
      {medium medium}
      {high medium} }

```

This means that

- Soft rules with low weight should use low effort.
- Soft rules with medium weight should use medium effort.
- Soft rules with high weight should use medium effort.

-enforce_voltage_areas off | strict | relaxed

Specifies the level of enforcement of voltage areas.

The definition for each mode is

- **off**

Voltage area constraints are turned off for routing.

- **strict**
Uses strict voltage area constraints for routing.
- **relaxed** (the default)
Uses relaxed voltage area constraints for routing.

-voltage_area_weight {{ *voltage_area_name weight* }...}

Specifies the degree to which routing is discouraged in non-native voltage areas for a given net. Valid weight values are **low**, **medium**, and **high**.

The native voltage areas for a net are the voltage areas that contain the ports connected to the net. The router always prefers to route a net in its native voltage area. When the router must route in non-native voltage areas, the higher the weight of the voltage area, the more the router tries to avoid routing in that voltage area.

If you do not specify a weight for a voltage area, it is assigned a default weight of **medium**.

This option is honored only if the **-enforce_voltage_areas** option is set to a value other than **off**.

-single_connection_to_pins *connection_mode*

Specifies whether the router can connect to a pin only once.

Valid values are **off**, **standard_cell_pins**, **macro_cell_pins**, **top_cell_pins**, **standard_and_macro_cell_pins**, **standard_and_top_cell_pins**, **macro_and_top_cell_pins**, **standard_and_macro_and_pad_cell_pins**, and **all_pins**. If the value is **off**, the router is allowed to connect to a pin any number of times. If the value is **all_pins**, all the pins in the design have the single-connection constraint and the router can connect to any pin only once. The other values specify the subset of pins that have the single-connection constraint.

The default is **off**, except for nets that use a nondefault width routing rule. For those nets, the value is always **all_pins**.

-mark_clock_nets_minor_change *true* | *false*

Controls whether changes on clock tree nets are limited during routing.

When **true** (the default), the router can make only limited changes to nets marked as clock tree nets in the database.

When **false**, nets marked as clock tree nets in the database are routed normally (the router can make any changes to these nets).

-track_auto_fill *true* | *false*

Specifies whether to fill empty space with routing tracks.

The default is **true**.

-tie_off_mode *all* | *rail_only*

Controls whether the router can connect tie-off nets to any power or ground (PG) structures, such as straps, rails, or pins, or only to a PG rail.

The definition for each mode is

- **all** (the default)
Tie-off nets can be connected to any PG structures.
- **rail_only**
Tie-off nets can be connected only to a PG rail.

-connect_within_pins_by_layer_name {{ *layer mode* }...}

Specifies, for each layer, whether to connect to pins by using vias or wires contained within the pin shapes. You must specify the

routing layers by using the layer names in the technology file.

This option can be set only when the top cell is open.

The valid mode values are

- **off** (the default)
Connect by using wires or vias anywhere on the pins.
- **via_standard_cell_pins**
Connect to standard cell pins by using vias contained within the pin shapes.
- **via_wire_standard_cell_pins**
Connect to standard cell pins by using vias and wires contained within the pin shapes.
- **via_all_pins**
Connect to any pins by using vias contained within the pin shapes.
- **via_wire_all_pins**
Connect to any pin by using vias and wires contained within the pin shapes.

-number_of_secondary_pg_pin_connections *int*

Specifies the maximum number of secondary power and ground (PG) pins in a cluster.

The definition of a cluster is a set of connected secondary PG pins. Each cluster has one connection with a PG strap or ring. If the value is larger than 0, a cluster should not contain more than the specified number of secondary PG pins.

By default, the setting is 0, which means that there is no maximum.

-separate_tie_off_from_secondary_pg *true* | *false*

Specifies whether to separate the tie-off connections from the secondary PG power connections.

By default (**false**), tie-off connections and secondary PG power connections are considered the same nets, so nondefault routing rules, minimum and maximum layer constraints, and other constraints on the net apply to both tie-off connections and secondary PG power connections.

When you separate the tie-off connections from the secondary PG power connections by setting this option to **true**, you can apply nondefault routing rules, minimum and maximum layer constraints, and other constraints only to the secondary PG power connections.

-default *true* | *false*

If **true**, this option resets all options to their default values.

The default is **false**.

-shielding_nets *pg_net_names*

Specifies the power and ground nets to be used for shielding by the **create_zrt_shield** command.

When routing signal nets, if the **-allow_pg_as_shield** option is **true** (the default), the router tries to route the shielded nets adjacent to any of the specified power and ground nets while considering other routing metrics such as congestion and wire length.

If the **-allow_pg_as_shield** option is **false**, this option has no effect.

If you do not specify this option, by default, the **create_zrt_shield** command uses the ground net for shielding.

You can also specify the power or ground net used for shielding by using the **-with_ground** option with the **create_zrt_shield** command; however, this option supports only a single power or ground net.

If you specify both the **-shielding_nets** option and the **create_zrt_shield -with_ground** option, the tool issues a warning message and uses the **-shielding_nets** setting.

-report_local_double_pattern_odd_cycles true | false

Controls whether local double pattern odd cycles are reported.

For double-patterning technology designs, a mask is split into two masks for lithography success. To check whether a mask can be split into two masks, the tool creates double-pattern links between any two polygons whose distance is smaller than the double-pattern spacing defined in technology file. A cycle is formed when a set of polygons is connected in a ring topology by double-pattern links.

A double-pattern odd cycle is a cycle that consists of an odd number of double-pattern links. Double-pattern odd cycles must be avoided or fixed because they prevent the mask from splitting into two masks. A local double-pattern odd cycle is a double-pattern odd cycle that is contained in a bounding box that is 20 global routing cells high by 20 global routing cells wide.

By default (**false**), Zroute does not report local double-pattern odd cycles.

When this option is **true**, Zroute reports local double-pattern odd cycles that are caused by signal detail routes. Zroute does not report local double-pattern cycles that contain only shapes that cannot be changed by Zroute. For example, Zroute does not report a local double-pattern odd cycle that contains only PG preroutes, pins, and standard-cell obstacles.

To check for local double-pattern odd cycles, use the **verify_zrt_route** command. This command guarantees the reporting of local double-pattern odd cycles when this option is **true**. Note that the **route_zrt_detail** command and other routing commands do not guarantee the reporting of double-pattern odd cycles.

-wire_on_grid_by_layer_name {{layer true | false}}...

Controls whether wires must be routed on-grid. This option controls the same functionality as the **onGrid** attribute for a metal layer in the technology file.

If you use this option, the tool ignores all **onGrid** or **onWireTrack** attributes defined in the technology file (both for metal layers and via layers). The default setting for any unspecified **onGrid** attributes is **false**; therefore, on-grid routing is required only for the metal layers specified with a setting of **true** in this option and via layers specified with a setting of **true** in the **-via_on_grid_by_layer_name** option.

-via_on_grid_by_layer_name {{layer true | false}}...

Controls whether the vias and their associated via enclosures must be routed on-grid. This option controls the same functionality as the **onGrid** attribute for a via layer in the technology file.

If you use this option, the tool ignores all **onGrid** or **onWireTrack** attributes defined in the technology file (both for metal layers and via layers). The default setting for any unspecified **onGrid** attributes is **false**; therefore, on-grid routing is required only for the via layers specified with a setting of **true** in this option and metal layers specified with a setting of **true** in the **-wire_on_grid_by_layer_name** option.

-min_shield_length_by_layer_name {{layer length}}...

Specifies the minimum shielded length in microns on each layer for the router to skip shielding the net wires with lengths less than these values. Since these wires are always unshielded, after shield creation of the net, when reporting shielding ratio in **create_zrt_shield** and **report_zrt_shield** commands, these wires are not counted.

The default length for each layer is 4-layer-pitch. Using -1 could set it back to the default value while a layer value has been set.

DESCRIPTION

The **set_route_zrt_common_options** command sets Zroute options that are common to global routing, track assignment, and detail

routing. The settings are saved with the design.

Note:

Hardcoded layer name strings are no longer supported. If the option settings saved with a design use hardcoded layer name strings, these option settings are not honored. You must reset these options using the layer names from the technology file instead of the hardcoded layer name strings.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the **-enforce_voltage_areas** option to **off**.

```
prompt> set_route_zrt_common_options -enforce_voltage_areas off
```

The following example sets the **-threshold_noise_ratio** option to 0.2.

```
prompt> set_route_zrt_common_options -threshold_noise_ratio 0.2
```

The following example sets the **-freeze_layer_by_layer_name** option to **true** for m3 and m4. Here "m3" and "m4" are the layer names in the technology file for metal3 and metal4.

```
prompt> set_route_zrt_common_options \  
-freeze_layer_by_layer_name {{m3 true} {m4 true}}
```

The following example sets the **-clock_topology** option to **comb**.

```
prompt> set_route_zrt_common_options -clock_topology comb
```

The following example sets the extra via cost multiplier to 0.5 on the VIA12 layer and to 1.5 on the VIA23 layer.

```
prompt> set_route_zrt_common_options \  
-extra_via_cost_multiplier_by_layer_name {{VIA12 0.5} {VIA23 1.5}}
```

If the technology file defines the VIA12 via layer between the M1 and M2 metal layers and the VIA23 via layer between the M2 and M3 metal layers, the following example sets the extra off-grid via cost multiplier on the VIA12 and VIA23 layers to 0.5. Note that you specify the metal layer, but the multiplier is set on the adjacent via layers.

```
prompt> set_route_zrt_common_options \  
-extra_via_off_grid_cost_multiplier_by_layer_name {{M2 0.5}}
```

SEE ALSO

get_route_zrt_common_options(2)
report_route_zrt_common_options(2)
set_ignored_layers(2)
set_net_routing_layer_constraints(2)

set_route_zrt_global_options

Sets the options for Zroute global routing.

SYNTAX

```
status set_route_zrt_global_options
[-exclude_blocked_gcells_from_congestion_report true | false]
[-layer_based_congestion_map true | false]
[-timing_driven true | false]
[-timing_driven_effort_level low | medium | high]
[-crosstalk_driven true | false]
[-double_pattern_utilization_by_layer_name {{layer int}...}]
[-effort minimum | low | medium | high | ultra]
[-force_full_effort true | false]
[-extra_blocked_layer_utilization_reduction int]
[-macro_boundary_track_utilization int]
[-macro_boundary_width int]
[-macro_corner_track_utilization int]
[-voltage_area_corner_track_utilization int]
[-default true | false]
[-auto_gcell true | false]
```

Data Types

int integer

ARGUMENTS

-exclude_blocked_gcells_from_congestion_report true | false

Controls whether blocked global routing cells (gcells) are included in the overflow percentage calculation.

By default (**false**), the total number of global routing cells is used to calculate the overflow percentages in the congestion report.

If the option is set to **true**, the overflow percentage calculation excludes the global routing cells that are blocked for routing. In cases where the global router has to route through blocked global routing cells to find a solution, those global routing cells are counted in the overflow number, but not in the total number.

-layer_based_congestion_map true | false

Controls whether layer-based congestion maps are created.

By default (**true**), both layer-based and aggregate congestion maps are created.

If the option is set to **false**, only an aggregate congestion map is created to reduce the storage requirements.

-timing_driven true | false

Enables (true) or disables (false) timing-driven global routing.

The default is false.

-timing_driven_effort_level low | medium | high

Specifies the effort level for the timing-driven flow. The effort levels trade off between timing QoR and DRC convergence, with the high effort level emphasizing timing QoR and the low effort level emphasizing DRC convergence.

This option takes effect only if the **-timing_driven** option is set to **true**.

The default is **high**.

-crosstalk_driven true | false

Enables (true) or disables (false) crosstalk-driven global routing.

The default is **false**.

-double_pattern_utilization_by_layer_name {{layer int}...}

Specifies the double pattern (DPT) utilization percentage for each layer. Specify the layers by using the layer names from the technology file.

The range for the utilization percentage is from 0 to 100.

The default is 80 for M1 and M2, 85 for M3, and 90 for any higher DPT layers. The value is 100 for non-DPT layers.

-effort minimum | low | medium | high | ultra

Specifies the global routing effort.

Valid values are

- **minimum**
Very fast run time. Runs a maximum of two phases.
- **low**
Fast run time. Runs a maximum of three phases.
- **medium** (default)
Medium run time. Runs a maximum of four phases.
- **high**
Slow run time, but better quality of results. Runs a maximum of five phases.
- **ultra**
Slower run time. Use on really congested designs. Runs a maximum of seven phases.

-force_full_effort true | false

Forces the maximum number of phases specified by the **-effort** option to try to achieve better results with longer runtime.

By default (**false**), the global router might stop before the maximum number of iterations, based on the incremental congestion reduction.

If set to **true**, the global router always performs the maximum number of routing phases, based on the specified effort level. For example, if this option is set to **true** and the **-effort** option is set to **high**, the global router always runs five routing phases.

-extra_blocked_layer_utilization_reduction int

Specifies the percentage of additional utilization to be reduced for areas where either the upper or lower layer is blocked and switching to the opposite layer would take more than one track resource.

The range is from 0 to 100.

The default is 0.

-macro_boundary_track_utilization *int*

Specifies the percentage of tracks to be used along macro boundaries.

The range is from 0 to 100.

The default is 100.

-macro_boundary_width *int*

Specifies the width of the macro boundary in terms of global route cells.

The range is from 0 to 10.

The default is 5.

-macro_corner_track_utilization *int*

Specifies the percentage of tracks to be used along macro corners.

The range is from 0 to 100.

The default is 100.

-voltage_area_corner_track_utilization *int*

Specifies the percentage of tracks to be used in the region within five global routing cells of a voltage area corner.

A smaller utilization reduces the number of global routes that can go through a global routing cell by artificially setting the congestion in that area higher. If the neighboring global routing cells have lower congestion, the routing might be pushed to them; however, this is a function of many other metrics in the global router.

This option has an effect only if the **enforce_voltage_areas** common route option is set to **strict**.

The range is from 0 to 100.

The default is 100.

-default true | false

Resets all options to their default values when set to true.

The default is false.

-auto_gcell true | false

When set to true, automatically increases gCell size when cell row height is too small. This option is default off as it is recommended for lower technology nodes.

The default is false.

DESCRIPTION

The **set_route_zrt_global_options** command sets the options for Zroute global routing. These settings affect all commands that perform Zroute global routing.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the **-timing_driven** option to **true**.

```
prompt> set_route_zrt_global_options -timing_driven true
```

The following example sets the **-double_pattern_utilization_by_layer_name** option on M1, M2, and M5.

```
prompt> set_route_zrt_global_options \  
-double_pattern_utilization_by_layer_name { {M1 85} {M2 85} {M5 70} }
```

The following example sets the **-auto_gcell** option to **true**.

```
prompt> set_route_zrt_global_options -auto_gcell true
```

SEE ALSO

report_route_zrt_global_options(2)

set_rp_group_options

Sets relative placement group attributes on the specified relative placement groups.

SYNTAX

```
collection set_rp_group_options
  rp_groups
  [-alignment bottom-left | bottom-pin | bottom-right]
  [-pin_align_name pin_name]
  [-utilization percentage]
  [-ignore]
  [-x_offset float]
  [-y_offset float]
  [-group_orient default | N | FN | S | FS]
  [-cell_orient_opt]
  [-auto_blockage]
  [-disable_buffering]
  [-cts_option fixed_placement | size_only]
  [-route_opt_option fixed_placement | in_place_size_only]
  [-psynopt_option fixed_placement | size_only | all_optimization]
  [-move_effort low | medium | high]
  [-allow_keepout_over_tapcell false | true]
  [-allow_non_rp_cells]
  [-place_around_fixed_cells standard | physical_only | all | none]
  [-anchor_corner bottom-left | bottom-right | top-left | top-right | rp-location [-anchor_row integer] [-anchor_column integer]]
  [-placement_type bit_slice | compression | vertical_compression]
  [-ignore_rows ignore_row_names_list]
  [-max_rp_width float]
  [-max_rp_height float]
```

Data Types

<i>rp_groups</i>	list or collection
<i>pin_name</i>	string
<i>percentage</i>	float
<i>float</i>	percentage
<i>ignore_row_names_list</i>	list of strings

ARGUMENTS

rp_groups
Specifies the relative placement groups for which you want to change the attributes. This is a required argument.

-alignment bottom-left | bottom-pin | bottom-right

Specifies the default alignment method to use when placing leaf cells and relative placement groups in the specified relative placement groups. If you do not specify this option, the tool uses bottom-left alignment.

You can override the default alignment method for a specific leaf cell or relative placement group when you add it to a relative placement group (by using the **add_to_rp_group** command).

-pin_align_name pin_name

Specifies the default alignment pin for the specified relative placement groups. During placement, the tool uses the alignment pin's location to align the cells within a column when you use the bottom-pin alignment type.

You can override the alignment pin for a specific leaf cell when you add the cell to the group (by using the **add_to_rp_group** command).

-utilization percentage

Specifies the utilization percentage to use for placement of the specified relative placement groups. Utilization is represented as a floating-point number between 0.0 (representing 0%) and 1.0 (the default, representing 100%).

-ignore

Indicates that the tool is to ignore the specified relative placement groups during placement and not treat them as relative placement groups. Instead, the tool places all of the group's parts individually, as it would normally do when there is no relative placement.

-x_offset float

Specifies a left coordinate (anchor) for the group, in microns, relative to the lower-left corner in the core area for top-level relative placement group. It is ignored for a lower-level relative placement group. Specifying only an x-offset allows a group to slide in the y-direction.

-y_offset float

Specifies a lower coordinate (anchor) for the group, in microns, relative to the lower-left corner in the core area for top-level relative placement group. It is ignored for a lower-level relative placement group. Specifying only a y-offset allows a group to slide in the x-direction.

-group_orient default | N | FN | S | FS

Specifies the orientation of a relative placement group.

If this value is not specified or is specified as **default** for any relative placement group, the tool chooses a suitable orientation for that relative placement group.

-cell_orient_opt

Applies automatic orientation on the cells in a relative placement group for optimizing wirelength.

-auto_blockage

Disallows buffer insertion inside the area of relative placement groups.

-disable_buffering

Disallows buffer insertion on nets within relative placement groups.

-cts_option fixed_placement | size_only

Specifies how to treat the cells in the relative placement group during clock tree synthesis and clock tree optimization.

If you specify **fixed_placement**, the cells in the relative placement group are treated as fixed during **compile_clock_tree**, **optimize_clock_tree**, and **clock_opt** activity and cannot be sized or moved.

If you specify **size_only**, the cells in the relative placement group can only be sized. This option is applies to the **compile_clock_tree**, **optimize_clock_tree**, and **clock_opt** commands.

This option applies only to top-level relative placement groups and is ignored for hierarchical relative placement groups. The top-level value is propagated to the hierarchical relative placement groups.

This option is not supported in Design Compiler topographical mode and will be ignored.

-route_opt_option fixed_placement | in_place_size_only

Specifies how to treat the cells in the relative placement group during **route_opt** and **psynopt on_route**.

If you specify **fixed_placement**, the cells in the relative placement group are treated as fixed during the **route_opt** and **psynopt - on_route** commands and cannot be sized or moved.

If you specify **in_place_size_only**, sizing changes are constrained for minimal engineering change order (ECO) placement changes. For example, a cell is sized to improve timing or design rule costs only if the newly-sized cell can fit into any available space adjacent to the original cell location. The resulting transformation is verified to ensure it is legal.

This option applies only to top-level relative placement groups and is ignored for hierarchical relative placement groups. The top-level value is propagated to the hierarchical relative placement groups.

This option is not supported in Design Compiler topographical mode and will be ignored.

-psynopt_option fixed_placement | size_only | all_optimization

Specifies how to treat the cells in the relative placement group during **psynopt** and **place_opt**.

If you specify **fixed_placement**, the cells in the relative placement group are treated as fixed during physical optimization and cannot be sized or moved. This value is not recommended for **place_opt**.

If you specify **size_only**, the cells in the relative placement group can only be sized.

If you specify **all_optimization**, all the preroute optimization capabilities can be applied to cells in the relative placement group. This might result in losing some cells from the relative placement group because cells can be decomposed or deleted from the netlist.

This option applies only to top-level relative placement groups and is ignored for hierarchical relative placement groups. The top-level value is propagated to the hierarchical relative placement groups.

This option is not supported in Design Compiler topographical mode and will be ignored.

-move_effort low | medium | high

Controls the extent to which a relative placement group can be moved from its initial location to preserve the relative placement without violating relative placement constraints.

The size of the region searched for placement of a relative placement group is progressively reduced as you reduce the effort from high to medium to low.

Coarse placement estimates an initial location for every top-level relative placement group. If those groups are placed at the same locations returned by coarse placement, the relative placement groups might need to be broken, which would violate the relative placement constraints for those groups.

Specifying high effort allows the higher movement of a relative placement group from its initial location in search of a location compared to low and medium effort, to maintain the relative placement constraints and potentially avoid breaking the relative placement group. The tradeoff for maintaining the relative placement group is a potential loss of QoR, because the group is moved from the location initially determined by coarse placement.

Specifying low effort indicates that the relative placement group is placed as close as possible to the initial location returned by coarse placement. The tradeoff for maintaining the location is a higher potential for breaking the relative placement groups and violating the relative placement constraints for some relative placement groups.

This option is not supported in Design Compiler topographical mode and will be ignored.

-allow_keepout_over_tapcell false | true

Specifies that the hard keepout in the relative placement group should be overlapped with tap cells if needed. By default, the value is false.

This option is not supported in Design Compiler topographical mode and will be ignored.

-allow_non_rp_cells

Specifies to allow nonrelative placement cells in the unused area of relative placement groups during place_opt and refine_placement.

-place_around_fixed_cells standard | physical_only | all | none

Specifies how to treat fixed cells in the floorplan during legalization of relative placement groups.

If you specify **standard**, the relative placement groups are legalized around fixed standard cells and the legalization engine avoids fixed physical-only cells.

If you specify **physical_only**, the relative placement groups are legalized around fixed physical-only cells, and the legalization engine avoids fixed standard cells.

If you specify **all**, the relative placement groups are legalized around both fixed standard and fixed physical-only cells.

If you specify **none**, the legalization engine avoids both fixed standard and fixed physical-only cells during legalization of relative placement groups.

The default of the option is **all**.

This option applies to top-level relative placement groups only and is ignored for hierarchical relative placement groups.

-anchor_corner bottom-left | bottom-right | top-left | top-right | rp-location

Specifies the corner for the anchor point that is set by using the **-x_offset** and **-y_offset** options.

If you specify **bottom-left** (the default), the anchor point corner is the lower-left corner of the relative placement group.

If you specify **bottom-right**, the anchor point corner is the lower-right corner of the relative placement group.

If you specify **top-left**, the anchor point corner is the upper-left corner of the relative placement group.

If you specify **top-right**, the anchor point corner is the upper-right corner of the relative placement group.

If you specify **rp-location**, the anchor point corner is the starting location of object at the row and column specified by the **-anchor_row** and **-anchor_column** options. When you specify **rp-location**, the **-anchor_row** and **-anchor_col** options are required and the specified location must not be empty.

This option applies to top-level relative placement groups only and is ignored for hierarchical relative placement groups. When you specify **-anchor_corner bottom-right**, the relative placement group is anchored at the bottom-right corner at the specified x- and y-coordinates during legalization.

When many blockages are present, a slight deviation from the anchor point might occur.

-anchor_row integer

Specifies the row of the object for which the **-anchor_corner rp-location** option is specified.

Note that this option is applicable only to the **-anchor_corner rp-location** option and is not accepted for other values of the **-anchor_corner** option.

-anchor_column *integer*

Specifies the column of the object for which the **-anchor_corner rp-location** option is specified.

Note that this option is applicable only to the **-anchor_corner rp-location** and is not accepted for other values of the **-anchor_corner** option.

-placement_type *bit_slice* | *compression* | *vertical_compression*

Specifies how to place cells in relative placement groups. The default is *bit_slice*.

When you specify the *bit_slice* keyword, the next row starts after the tallest object in the row and the next column starts after the widest object in the column. Both the row and column alignments are preserved.

When you specify the *vertical_compression* keyword, columns are compressed such that no gap is between leaf cells, lower-level hierarchical relative placement groups, and keepouts in a column. Row alignment is not preserved.

When you specify the *compression* keyword, rows are compressed such that no gap is between leaf cells, lower-level hierarchical relative placement groups, and keepouts in a row. Column alignment is not preserved. If you set the **-alignment** option to bottom-right or bottom-pin and set this option to *compression*, then compression is ignored and alignment is respected.

If both the **-utilization** and **-placement_type compression** options are specified, the utilization constraints are observed with gaps between leaf elements in a relative placement row.

The setting of this option does not propagate to child relative placement groups.

-ignore_rows *ignore_row_names_list*

Specifies the site rows that must be ignored for the relative placement group cells.

To ignore site rows during relative placement, you must first define the **rp_ignore_row_name** attribute on the site rows to be ignored. The *ignore_row_names_list* argument is a list that includes the **rp_ignore_row_name** attribute value for each site row to be ignored.

This option applies only to top-level relative placement groups and is ignored for hierarchical relative placement groups.

-max_rp_width *float*

Specifies the maximum allowable width in microns for the top-level relative placement groups. It is ignored for the lower-level relative placement groups. If you specify this option, the tool tries to restrict the width of the top-level relative placement groups to the specified maximum value during **psynopt**, **place_opt** and **route_opt**.

-max_rp_height *float*

Specifies the maximum allowable height in microns for the top-level relative placement groups. It is ignored for the lower-level relative placement groups. If you specify this option, the tool tries to restrict the height of the top-level relative placement groups to the specified maximum value during **psynopt**, **place_opt** and **route_opt**.

DESCRIPTION

The **set_rp_group_options** command sets the attributes of the specified relative placement groups. The same attributes can be set during the creation of the relative placement groups by using the **create_rp_group** command. This command is supported only for designs that do not contain multiply-instantiated designs.

This command returns a collection containing the relative placement groups for which the attributes have changed. If attributes have not changed for any object, the empty string is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **set_rp_group_options** command to change the attributes for a relative placement group.

```
prompt> set_rp_group_options ripple::grp_ripple \  
-utilization 0.95 -ignore  
{ripple::grp_ripple}  
  
prompt> define_user_attribute \  
-type string -class site_row rp_ignore_row_name  
prompt> set_attribute [get_site_rows STD_ROW_112] \  
rp_ignore_row_name clock  
prompt> set_rp_group_options d::rp -ignore_rows {clock}
```

SEE ALSO

- add_to_rp_group(2)
- all_rp_groups(2)
- create_rp_group(2)
- get_rp_groups(2)
- remove_from_rp_group(2)
- remove_rp_groups(2)
- report_rp_group_options(2)
- write_rp_groups(2)

set_rtl_load

Sets an RTL load value for capacitance and resistance on pins, ports, and nets.

SYNTAX

```
status set_rtl_load
  [-min]
  [-max]
  pin_net_list
  [-capacitance cvalue]
  [-resistance rvalue]
```

Data Types

<i>pin_net_list</i>	list
<i>cvalue</i>	float
<i>rvalue</i>	float

ARGUMENTS

-min

Specifies that the load value is to be used for minimum delay analysis. If no minimum load value is specified, the maximum value is used. If neither **-min** nor **-max** are specified, the load values are used for both minimum and maximum delay calculations.

-max

Specifies that the load value is to be used for maximum delay analysis. If no value is specified for maximum analysis on a net, the minimum value is used. If neither **-min** nor **-max** are specified, the load values are used for both minimum and maximum delay calculations.

pin_net_list

Specifies a list of ports, pins, and nets in the current design on which the RTL loads are set.

-capacitance *cvalue*

Specifies the capacitance value with which to set the RTL load value on the pins, ports, and nets contained in *pin_net_list*. The *cvalue* must be expressed in units consistent with the technology library used during optimization. For example, if the technology library specifies capacitance values in picofarads, *cvalue* must also be expressed in picofarads. If *cvalue* is not specified, it remains unchanged from any previous setting. If *cvalue* is never set, the value defaults to zero. You must specify either **-capacitance** or **-resistance**, and you can specify both.

When *cvalue* is set on a net, the capacitance is split equally among the sticky pins of that net. For more information about sticky pins, see the "Sticky Pins and RTL Load on Nets" subsection below.

-resistance *rvalue*

Specifies the resistance value with which to set the RTL load value on the ports, nets, or pins contained in *pin_net_list*. The *rvalue* must be expressed in units consistent with the technology library used during optimization. For example, if the technology library specifies resistance values in Kohms, *rvalue* must also be expressed in Kohms. If *rvalue* is not specified, it remains unchanged from any previous setting. If *rvalue* is never set, the value is calculated based on the specified capacitance value. For more information, see the "Calculation of Wire Delays and Capacitances" subsection below. You must specify either -**capacitance** or -**resistance**, and you can specify both.

When *rvalue* is set on a net, the resistance value is assigned to all sticky pins of that net. This is not the same method used to distribute *cvalue* on a net. For more information about sticky pins, see the "Sticky Pins and RTL Load on Nets" subsection below.

DESCRIPTION

The **set_rtl_load** command overrides the calculation of wire characteristics during RTL level synthesis and other early phases of synthesis. By default, the **compile** command uses wire load models (WLMs) to calculate both the total capacitance and the time-of-flight delay of a wire. When **set_rtl_load** values are specified for particular pins, ports, or nets, the default WLM calculation is replaced by a detailed RC calculation.

Apply **set_rtl_load** only to sticky pins, or nets attached to sticky pins. For information about sticky pins, see the "Sticky Pins and RTL Load on Nets" subsection below.

The **set_rtl_load** command values are ignored during the later stages of synthesis that include the **reoptimize_design** command, because the placement-based wire capacitance and delay calculations of these commands are more accurate than the **set_rtl_load** values.

Sticky Pins and RTL Load on Nets

Annotating capacitance and resistance values on a design at the RTL level can be difficult. The objects to which you want to annotate might not exist before the first compile. Moreover, compile optimization could change or delete an object you want to annotate. The **set_rtl_load** command circumvents this problem by defining the concept of "sticky pins". Sticky pins are pins (or ports) of your design that are guaranteed to be persistent throughout the early stages of optimization. Therefore, annotated capacitance and resistance can stick to these pins and not be lost. You can apply **set_rtl_load** only to sticky pins.

A sticky pin is defined as one of the following:

- A top-level port of your design
- A pin of a hierarchical block
- A pin of a cell marked with the **dont_touch** attribute
- A pin of a cell marked with the **size_only** attribute
- A pin of a black-box cell (has a 'b' attribute in **report_cell**)
- A pin connected to a net marked with the **dont_touch** attribute.

Applying **set_rtl_load** to a net is shorthand for applying **set_rtl_load** to the sticky pins of the net. For example, consider a net that is connected to a top-level port, the pin of a RAM that is a black box, two hierarchical pins, and two leaf cells that are not marked with the **dont_touch** attribute. This net is connected to four sticky pins and two non-sticky pins. If the specified RTL load capacitance for the net is 4, the capacitance is distributed equally, with 1 to each of the four sticky pins.

If an RTL load resistance is specified for the net, that resistance is assigned to each of the four sticky pins. For example, if the specified resistance is 3, each of the sticky pins is assigned an RTL load resistance of 3.

If you want more detailed control of the assignment of RTL load capacitance and resistance on a net, assign values to each pin individually.

Calculation of Wire Delays and Capacitances

When **set_rtl_load** is in use, wire load models are switched off for the annotated wire. Instead, the values are calculated with a hybrid of annotated RTL loads and wire load models.

The total capacitance of a wire is needed in the delay calculation of the wire's driving cell. The wire capacitance is calculated as the sum of all annotated RTL load capacitances of the wire, plus an extra capacitance to account for the wiring required to connect to any nonannotated pin. For the example net with four annotated sticky pins, two hierarchy blocks, and two other leaf pins, the wire's total capacitance is the sum of the following:

- The total annotated RTL load capacitance on the four sticky pins
- The WLM calculation of the wire capacitance needed to connect the two nonsticky pins
- The WLM calculation of the wire capacitance needed to connect any nonannotated pins within the subhierarchy of the top-level block.

One implication of the method by which capacitances are calculated is that setting an RTL load capacitance is not strictly an additive process. If a net's capacitance as calculated by wire load models is 2.0, and you set an RTL load capacitance of 1.0 on the net, the resulting capacitance could be more or less than 3.0. If you intend to use **set_rtl_load** to obtain a particular fixed capacitive value, see the "Layout-based Annotation" subsection below.

The time-of-flight delay on a wire that has RTL loads is calculated in two phases: first, an RC network that represents the wire is built; second, Elmore delay calculation is applied to the RC network. The RC network is built as follows:

- Any leaf-level pin or top-level port that has an annotated RTL load is given an RC segment that is connected to that pin or port. The capacitance and resistance values of the RC segment are taken from the **set_rtl_load** values.
- Any hierarchy pin that has an annotated RTL load is given an RC segment. The capacitance and resistance values of the RC segment are taken from the **set_rtl_load** values. Wire segments within the hierarchy block are placed on one side of the pin's RC segment. Wire segments outside the hierarchy block are placed on the other side of the pin's RC segment. In this way, annotated hierarchy pins segment the wire into two parts.
- Any leaf-level pin that does not have an annotated RTL load is connected into the RC network at the appropriate hierarchy level. The capacitance and resistance of the connection is calculated using the wire load model for that hierarchy level. For example, if 3 nonannotated pins are connected to the network at the top level, the connection capacitance and resistance are calculated using the top-level wire load model for 3 connections.

Default Resistance Values

It is not necessary to specify resistance values directly to the **set_rtl_load** command. Instead, **set_rtl_load** can calculate the resistance value as a constant factor times capacitance. You determine the constant factor by setting the **rtl_load_resistance_factor** variable to the required value. The default is 0.0. For example, if you set the **rtl_load_resistance_factor** variable to 0.5, specify the RTL load capacitance of a pin as 4, and do not specify the RTL load resistance, **set_rtl_load** calculates the RTL load resistance to be 2. The factor is applied to each annotated pin of a net individually, not to the net's capacitance as a whole. The default library units are used throughout.

Reporting and Resetting RTL Load Values

Use one of the following to display a report of the current value of RTL load values on cells or nets:

```
prompt> report_cell -connections cells
```

```
prompt> report_net -connections -verbose nets
```

Use the **write_rtl_load** command to display all RTL load information in the current design.

Remove RTL load annotations using the **remove_rtl_load** command. The **reset_design** command removes all attributes from a design, including load annotations placed by **set_rtl_load**.

Layout-based Annotation

RTL loads are intended to be used for annotation of physical interconnect information, such as RC values derived from top-level floorplans. Unlike the traditional layout-based annotation commands (**set_load** for nets, **read_sdf** and **set_annotated_delay**), RTL loads can be used at the RTL level and for preplaced designs. However, RTL loads are not a replacement for the traditional layout-based commands. The **reoptimize_design** command has sophisticated algorithms to incrementally update annotated capacitances and delays from the traditional layout-based commands as layout-based optimization proceeds. RTL load values are fixed. Thus, in the **reoptimize_design** and **report_timing** commands, annotations from **set_load** for nets, **read_sdf** and **set_annotated_delay** override any calculation based on RTL loads.

It is possible that your design methodology can generate early place and route results that are available before your RTL synthesis is finalized. Use current information from the layout to obtain RTL load annotation for the next pass of RTL synthesis using the **calculate_rtl_load** command. The **calculate_rtl_load** command converts back-annotated layout information from **set_load** for nets, **read_sdf**, and **set_annotated_delay** to RTL load values suitable for RTL level optimization. A typical sequence of commands is shown in the following script example:

```
/* Load your currently placed netlist */
read my_netlist.db
/* Annotate capacitance and delay information from placement */
read_sdf my_delays.sdf
include my_set_load_info.scr

/* Generate RTL load information for selected nets
 * and save the result
 */
calculate_rtl_load -cap -delay [get_nets *]
calculate_rtl_load -cap -delay [get_pins my_ram/*]
write_rtl_load -output my_rtl_load.scr

/* Apply the RTL loads to the RTL design and compile */
remove_design -designs
read my_rtl.db
include my_rtl_load.scr
compile
```

For more information, see the man page for the **calculate_rtl_load** command.

There are some caveats associated with the use of **calculate_rtl_load**. Use **set_rtl_load** and **calculate_rtl_load** only on nets that are predictably long. This includes nets that cross between floorplan blocks or nets that connect to large macro blocks. Do not use **set_rtl_load** and **calculate_rtl_load** for nets that occur deep in a physical hierarchy. Unlike the traditional layout-based annotation commands, which are kept up-to-date by **reoptimize_design**, RTL loads do not change if the layout changes. Deep in the physical hierarchy, a net that is long in one iteration of placement is not guaranteed to be long in the next iteration. Synthesis results could degrade if unreliable RTL loads are annotated onto the design.

EXAMPLES

The following example sets an RTL load capacitance of 4 on each of the specified pins and ports. The RTL load resistance is left as its default value.

```
prompt> set_rtl_load -cap 4 {my_port my_ram/ADDR[*]}
```

The following example sets maximum and minimum RTL load values on a pin:

```
prompt> set_rtl_load -max -cap 4 -res 0.5 my_hier/SIG1
```

```
prompt> set_rtl_load -min -cap 3 -res 0.0 my_hier/SIG1
```

The following example splits an RTL load capacitance of *4* across the sticky pins of a net, then assigns a resistance of *0.5* to each sticky pin.

```
prompt> set_rtl_load -cap 4 my_hier/net1
```

```
prompt> set_rtl_load -res 0.5 my_hier/net1
```

SEE ALSO

- `remove_rtl_load(2)`
- `report_cell(2)`
- `report_net(2)`
- `reset_design(2)`
- `set_dont_touch(2)`
- `set_size_only(2)`
- `set_wire_load_model(2)`
- `write_rtl_load(2)`
- `rtl_load_resistance_factor(3)`

set_scaling_lib_group

Specifies the scaling library group (previously defined by the **define_scaling_lib_group** command) to use for the current design or a subdesign.

SYNTAX

```
status set_scaling_lib_group  
  [-min min_group]  
  [-max max_group]  
  [-object_list objects]  
  [group]
```

Data Types

```
min_group  string  
max_group  string  
objects    collection  
group      string
```

ARGUMENTS

-min *min_group*

Specifies the scaling library group to use for minimum delay analysis. If you do not specify a scaling library group for minimum delay analysis, the maximum group is used. The **-min** option cannot be used without the **-max** option. If neither of these options are specified, the library group applies to both minimum and maximum delay analysis.

-max *max_group*

Specifies the scaling library group to use for maximum delay analysis. If this option is not specified, the scaling group is used for both maximum and minimum delay analysis.

-object_list *objects*

Specifies the cells or top-level ports on which to apply the scaling library group. If you do not use this option, the scaling library group applies to the top-level design. This option accepts both leaf-level cells and hierarchical blocks.

group

Specifies the scaling library group to use for both maximum and minimum delay analysis.

DESCRIPTION

The **set_scaling_lib_group** command set a scaling library group on the specified design objects. The tool performs interpolation between libraries in this group for voltage, temperature, or both voltage and temperature. You can set scaling library for maximum, minimum, or both maximum and minimum delay analysis. You can set multiple scaling library groups on an object.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example defines a scaling library group named g1 and applies it to the top/inv cell.

```
prompt> define_scaling_lib_group -name g1 { 0.9.db 1.1.db 1.3.db }  
prompt> set_scaling_lib_group -object_list top/inv g1
```

SEE ALSO

- define_scaling_lib_group(2)
- remove_scaling_lib_group(2)
- create_scenario(2)
- set_operating_conditions(2)

set_scan_compression_configuration

Specifies the scan compression configuration for the design.

SYNTAX

```
status set_scan_compression_configuration
  [-chain_count chain_count
  | -max_length max_chain_length
  | -minimum_compression compression_factor]
  [-xtolerance high | default]
  [-synchronize_chains none | all | tail | head]
  [-compressor_pipeline false | true]
  [-serialize chip_level | core_level | none]
  [-integration_only true | false]
  [-hybrid true | false]
  [-force_diagnosis true | false]
  [-static_x_chain_isolation true | false]
  [-inputs number_of_inputs]
  [-outputs number_of_outputs]
  [-base_mode base_mode_name]
  [-test_mode scan_compression_mode_name]
  [-min_power false | true]
  [-shared_inputs number_of_shared_inputs]
  [-shared_outputs number_of_shared_outputs]
  [-identical_cores core_list]
  [-scramble_identical_outputs true | false]
  [-shared_block_select false | true]
  [-shared_codec_controls false | true]
  [-shift_power_groups false | true]
  [-shift_power_chain_length chain_length
  | -shift_power_chain_ratio chain_ratio]
  [-shift_power_clock clock_name]
  [-shift_power_disable test_control_name]
  [-location compressor_decompressor_location]
```

Data Types

<i>chain_count</i>	integer
<i>max_chain_length</i>	integer
<i>compression_factor</i>	integer
<i>number_of_inputs</i>	integer
<i>number_of_outputs</i>	integer
<i>base_mode_name</i>	string
<i>scan_compression_mode_name</i>	string
<i>number_of_shared_inputs</i>	integer
<i>number_of_shared_outputs</i>	integer
<i>core_list</i>	string

<i>chain_length</i>	integer
<i>chain_ratio</i>	integer
<i>clock_name</i>	string
<i>test_control_name</i>	string
<i>compressor_decompressor_location</i>	string

ARGUMENTS

-chain_count *chain_count*

Specifies the number of compressed scan chains to build. The tool attempts to meet this constraint. If unable to do so, the tool issues a warning. See the DESCRIPTION section for more information.

-max_length *max_chain_length*

Specifies the maximum allowed length of the compressed scan chains. The tool attempts to meet this constraint. If unable to do so, the tool issues a warning. See the DESCRIPTION section for more information.

-minimum_compression *compression_factor*

Specifies the minimum required amount of compression, relative to the underlying standard scan base mode chain count. This is a relative method of specifying the compressed scan chain count. The standard scan chains are subdivided into compressed scan chains according to this ratio, along with a 20 percent pattern inflation factor to account for compression overhead:

$$\text{num\_compressed\_chains} = \text{num\_standard\_chains} * \text{compression\_factor} * 1.2$$

The minimum value is 2. The default is 10. See the DESCRIPTION section for more information.

-xtolerance *high | default*

Enables the insertion of fully X-tolerant scan compression structures when the value is set to **high**. When set to **default** (the default), the tool does not insert fully X-tolerant scan compression structures.

For information on high X-tolerance scan-in and scan-out pin requirements, see the TEST-1722 man page.

-synchronize_chains *none | all | tail | head*

Specifies the shift state of all compression mode chains. The values are as follows:

- **head** specifies that the first shift state of all compression mode chains are synchronized to the same clock edge.
- **tail** specifies that the last shift state of all compression mode chains are synchronized to the same clock edge.
- **all** specifies that both the first and the last shift states of all the compression mode chains are synchronized.
- **none** (the default behavior) specifies that no synchronization is performed.

-compressor_pipeline *false | true*

Enables compressor input pipelining.

This option enables pipeline registers at the compressor inputs (between the compressed scan chains and the compressor logic). These pipeline registers can push any partial-cycle paths and long routes to the input side of these pipeline registers so that a full clock cycle is available for the compressor XOR logic.

This feature requires that tail scan data pipelining registers also be used, which are enabled by the **-tail_pipeline_stages** option of the **set_pipeline_scan_data_configuration** command) The compressor pipeline registers do not count against the tail scan

data pipeline register depth.

The default is **false**, which disables the feature.

-serialize chip_level | core_level | none

This option enables the insertion of the serializer when the value is set to **chip_level** or **core_level**. The value **chip_level** inserts both the serializer and the serializer clock controller at the top level. The value **core_level** inserts the serializer but not the clock controller, and should be specified in hierarchical adaptive-scan synthesis (HASS) flows. The value **none**, which is the default, inserts neither the serializer nor the clock controller.

-integration_only true | false

Enables scan compression integration. The design must have cores that have scan compression structures inserted. This feature integrates the cores at the chip level by connecting the test signals to chip-level ports and creating a chip-level test protocol. The default of this option is **false**.

This option is mutually exclusive with the **-hybrid** option.

-hybrid true | false

When set to **true**, turns on the hybrid flow. This enables insertion of scan compression logic at the chip level along with integration of cores with scan compression structures in one pass. The default of this option is **false**.

This option is mutually exclusive with the **-integration_only** option.

-force_diagnosis true | false

When set to **true**, issues an error if a compressor cannot be built with full diagnosis capabilities. The default of this option is **false**.

-static_x_chain_isolation true | false

When set to **true**, specifies that cells that frequently capture X values are to be isolated into separate scan chains, called X chains. These X chains are observed only in direct observability modes and excluded from the full observation mode. The default of this option is **false**.

This feature requires that the following options also be set:

```
set_dft_drc_configuration -static_x_analysis enable
set_scan_compression_configuration -xtolerance high
```

-inputs number_of_inputs

Specifies the number of scan inputs to load scan data for scan compression. The tool reuses the scan inputs from the underlying standard scan base mode before creating new scan inputs. The default is equal to the minimum number of chains that DFT would build if scan compression were not enabled.

-outputs number_of_outputs

Specifies the number of scan outputs to observe scan data for scan compression. The tool reuses the scan outputs from the underlying standard scan base mode before creating new scan outputs. The default is equal to the minimum number of chains that DFT would build if scan compression were not enabled.

-base_mode base_mode_name

Specifies the standard scan base mode that is to be associated with the scan compression mode indicated by the **-test_mode** option. The default is **Internal_scan**.

-test_mode scan_compression_mode_name

Specifies the scan compression test mode to which the specification applies. The default is *all_dft*, except when **define_test_mode** is used. In this case, the default is the value of the last *test_mode* defined.

-min_power false | true

When set to **true**, specifies that compressor inputs are to be gated for power saving. The **insert_dft** command creates the compressor logic such that it is active only during scan shift in that compression mode. The compressor logic is inactive during scan capture in that mode, during other test modes, and during mission mode. This eliminates power consumption due to switching activity in the compressor during functional operation (which is at-speed rather than at test clock frequencies). The default of this option is **false**.

-shared_inputs number_of_shared_inputs

Specifies the number of top-level scan inputs to share across all codecs in a HASS or Hybrid integration flow. The minimum allowed value is the widest input width across all codecs. The maximum allowed value is the sum of all codec input widths, which disables codec I/O sharing. By default, codec I/O sharing is disabled.

If high X-tolerance is used, additional scan inputs are required. The number of additional scan-in pins is equal to log2 of the number of shared codecs, rounded up to the next integer value.

-shared_outputs number_of_shared_outputs

Specifies the number of top-level scan outputs to share across all codecs in a HASS or Hybrid integration flow. The minimum allowed value is the widest output width across all codecs. The maximum allowed value is the sum of all codec output widths, which disables output sharing. By default, the outputs are fully shared when codec I/O sharing is enabled by the **-shared_inputs** option.

The **-shared_outputs** option by itself does not enable codec I/O sharing; the feature is only enabled if the **-shared_inputs** option is also specified.

-identical_cores core_list

Specifies the list of identical core instances in the current design. This information allows the shared codec I/O feature to optimize their scan-in and scan-out connections for improved diagnosability. Wildcards and collections are supported.

If you have multiple groups of identical cores, provide all of them as a flat list; the tool determines the grouping automatically by comparing their codec and compressed scan chain characteristics.

This option requires that shared codec I/O be enabled with the **-shared_inputs** option.

-scramble_identical_outputs true | false

When set to **true**, the tool creates nonidentical (scrambled) connections for identical cores, which improves the diagnosability of identical cores. By default, the tool creates identical output connections for identical cores.

This option requires that identical cores be defined with the **-identical_cores** option.

-shared_block_select false | true

Specifies whether to use shared or dedicated block-select signal connections for identical shared-I/O cores that have dedicated outputs.

When set to **true**, the tool uses shared block-select connections for the cores. The default is **false**, which uses dedicated (separate) block-select signals for the cores.

Enabling this option requires the following:

- Shared codec I/O is enabled with the **-shared_inputs** option.
- The cores contain shared-I/O codecs with high X-tolerance, such that the core has one or more block-select signals.
- All cores in the current design or partition are identical instances of this core, specified with the **-identical_cores** option.
- Dedicated (unshared) outputs are used for the cores, specified with the **-shared_outputs** option.

-shared_codec_controls false | true

When set to **true**, the tool adds gating logic at the inputs of the output sharing compressor to selectively disable shared codecs from being observed. You can then use the **-disable_codecs** option of the **write_test_protocol** command to write a test protocol that excludes one or more codecs.

If you are using IEEE 1500 test-mode control, the **write_test_protocol** command automatically creates a test protocol that disables the codecs. Otherwise, you are responsible for test protocol and/or netlist modifications to disable the codecs.

The default is **false**, which does not add any gating logic.

This option requires that shared codec I/O be enabled with the **-shared_inputs** option.

-shift_power_groups false | true

Enables the shift power groups feature. This feature implements a shift power control chain that allows ATPG to selectively gate groups of compressed chains for shift power reduction. The gated chains still shift, but they shift constant (non-toggling) values into the chains, which reduces toggle activity.

When this feature is enabled, the following options are also required:

- **-shift_power_chain_length** or **-shift_power_chain_ratio**
- **-shift_power_clock**

The default is **false**, which does not implement the control chain or gating logic.

-shift_power_chain_length *chain_length*

Specifies the length of the shift power control (SPC) chain. This value directly controls the number of compressed chain groups.

Use this option to keep the length of the SPC chain constant as the compression architecture changes.

-shift_power_chain_ratio *chain_ratio*

Specifies the ratio of compressed chains to each shift power control (SPC) register. This value directly controls the number of compressed chains in each group.

Use this option to keep the group size constant as the compression architecture changes.

-shift_power_clock *clock_name*

Specifies the clock to use for the shift power control chain. The specified clock must be previously declared as a test clock using the **set_dft_signal** command. This option is required when using shift power groups.

-shift_power_disable *test_control_name*

Specifies the signal that, when asserted to its active value, disables the shift power logic. The specified signal must be previously declared as a TestControl signal using the **set_dft_signal** command. By default, the shift power logic is always active and no shift power disabling logic is implemented.

-location *compressor_decompressor_location*

Specifies the instance name in which the compressor and decompressor will be instantiated. The instance name cannot be a library cell, a black-box CTL model, or a black box.

DESCRIPTION

The **set_scan_compression_configuration** command allows you to specify certain DFTMAX insertion parameters prior to using the **insert_dft** command. The parameters include scan chain specifications such as compressed scan chain count, the number of scan-in and scan-out pins, the X-tolerance capabilities, and core integration strategies.

There are three related options to control the number of compressed scan chains. Only one option can take effect at a time. They are, in order of highest precedence first:

- **-max_length**
- **-chain_count**
- **-minimum_compression**

You can use these options to directly or indirectly specify the scan compression ratio for your design. If none of these options are specified, a minimum compression value of 10 (relative to the underlying standard scan chain count) is used. The maximum compressed chain count is 32000.

If you are using high X-tolerance, the maximum compressed chain count might be lower. For more information, see the TEST-1722 man page.

There are five related options to configure the shared codec I/O feature. They are:

- **-shared_inputs**
- **-shared_outputs**
- **-identical_cores**
- **-scramble_identical_outputs**
- **-shared_codec_controls**

There are five related options to configure shift power groups. They are:

- **-shift_power_groups**
- **-shift_power_chain_length**
- **-shift_power_chain_ratio**
- **-shift_power_clock**
- **-shift_power_disable**

SEE ALSO

define_test_mode(2)
insert_dft(2)
preview_dft(2)
report_scan_compression_configuration(2)
reset_scan_compression_configuration(2)
set_pipeline_scan_data_configuration(2)

set_scan_configuration

Specifies the scan chain design.

SYNTAX

```
int set_scan_configuration
  [-chain_count chain_count
  | -max_length max_chain_length
  | -exact_length chain_length
  | -count_per_domain chain_count]
  [-add_lockup true | false]
  [-clock_mixing no_mix | mix_edges | mix_clocks | mix_clocks_not_edges]
  [-add_test_retiming_flops begin_and_end | begin_only | end_only | none]
  [-create_dedicated_scan_out_ports true | false]
  [-internal_clocks none | single | multi]
  [-insert_terminal_lockup true | false]
  [-lockup_type latch | flip_flop]
  [-mix_internal_clock_driver true | false]
  [-preserve_multibit_segment false | true]
  [-style multiplexed_flip_flop | clocked_scan | lssd | combinational | scan_enabled_lssd | none]
  [-shared_scan_in pin_count]
  [-exclude_elements exclude_list]
  [-voltage_mixing true | false]
  [-power_domain_mixing true | false]
  [-test_mode mode_name]
  [-domain_based_scan_enable true | false]
  [-reuse_mv_cells true | false]
  [-pipeline_scan_enable true | false]
  [-pipeline_fanout_limit max_scan_cells]
  [-create_test_clocks_by_system_clock_domain true | false]
  [-replace true | false]
  [-hierarchical_isolation true | false]
  [-static_x_chain_isolation X_chains_isolation]
```

Data Types

```
chain_count      integer
max_chain_length integer
chain_length    integer
pin_count       integer
exclude_list    list
max_scan_cells  integer
```

ARGUMENTS

-chain_count *chain_count*

Specifies the number of scan chains to build. The tool attempts to meet this constraint. If unable to do so, the tool issues a warning. See the DESCRIPTION section for more information.

-max_length *max_chain_length*

Specifies the maximum allowed length of the scan chains. The tool attempts to meet this constraint. If unable to do so, the tool issues a warning. See the DESCRIPTION section for more information.

-exact_length *chain_length*

Specifies a positive integer that indicates the exact chain length. **insert_dft** builds, as much as possible, scan chains with the specified length. See the DESCRIPTION section for more information.

-count_per_domain *chain_count*

Specifies a positive integer for the number of chains that **insert_dft** is to build per clock domain. See the DESCRIPTION section for more information.

-add_lockup true | false

When set to its default of **true**, inserts lockup latches (synchronization element) between clock domain boundaries on scan chains. To disable synchronization element insertion, set this option to **false**. If the scan specification does not mix clocks on chains, **insert_dft** ignores this option.

-clock_mixing no_mix | mix_edges | mix_clocks | mix_clocks_not_edges

Specifies whether the **insert_dft** command can include cells from different clock domains in the same scan chain. The following allowed values control the clocking of cells in scan chains to be inserted by the **insert_dft** command:

- **no_mix** (the default)
Cells must be clocked by the same edge of the same clock.
- **mix_edges**
Cells must be clocked by the same clock, but the clock edges can be different.
- **mix_clocks_not_edges**
Cells must be clocked by the same clock edge, but the clocks can be different.
- **mix_clocks**
Cells can be clocked by different clocks and different clock edges.

-add_test_retiming_flops begin_and_end | begin_only | end_only | none

Specifies whether the **insert_dft** command can insert leading edge retiming lockup flip-flops to scan chains clocked with trailing edge scan elements. If a scan chain contains only leading edge scan elements, a retiming lockup flip-flop is not inserted. The following allowed values control the lockup flip-flop insertion by the **insert_dft** command:

- **begin_and_end**
Retiming lockup flip-flops are added to the scan chain beginning and end.
- **begin_only**
Retiming lockup flip-flop is added only to the scan chain beginning.
- **end_only**
Retiming lockup flip-flop is added only to the scan chain end.
- **none** (the default)
No retiming flip-flop is inserted.

-create_dedicated_scan_out_ports true | false

When set to **true**, instructs **insert_dft** to implement dedicated scan-out signal ports on the current design. When set to **false** (the default), **insert_dft** uses mission-mode ports as scan-out ports whenever possible.

-internal_clocks none | single | multi

This option applies only to the multiplexed flip-flop scan style; it is ignored for other scan styles.

The **-internal_clocks** option can be set to the following values:

- **single**
The **insert_dft** command treats internal clock network regions driven by any combinational gate, including buffers and inverters, as separate clocks for the purposes of scan chain architecture.
- **none**
The **insert_dft** command does not perform internal clock analysis. This is the default.
- **multi**
The **insert_dft** command treats internal clock network regions driven by multiple-input gates, such as MUX cells, as separate clocks for the purposes of scan chain architecture. Buffers and inverters are transparent for the analysis.

Integrated clock-gating cells are transparent for the determination of internal clocks.

The **-internal_clocks** setting is common to all test modes. As a result, this option can be specified only if the **-test_mode** option value is set to 'all'. Otherwise, the tool issues a warning message, ignores the **-internal_clocks** option, and processes the remaining **set_scan_configuration** options.

-insert_terminal_lockup true | false

When set to **true**, inserts synchronization element at the end of scan chains. The default is **false**. This option applies only to the multiplexed flip-flop scan style; it is ignored for other scan styles.

-lockup_type latch | flip_flop

Selects the type of synchronization element used in the scan chain. The default lock-up type is a level-sensitive latch. If you choose **flip_flop** as the lock-up type, an edge-triggered flip-flop is used as the synchronization element. You can use this option in conjunction with the **-add_lockup** option or **-insert_terminal_lockup** options.

-mix_internal_clock_driver true | false

When set to **true**, **insert_dft** allows mixing in a scan chain of any internal clocks that are derived from a single external clock pin. This capability can only be activated when the clock mixing scan configuration is either **no_mix** or **mix_edges**.

-preserve_multibit_segment false | true

Specifies whether to treat sequential multibit components as scan segments. A multibit component is a group of cells with identical functionality, inferred from RTL code or created by the **create_multibit** command. Depending on tool configuration and design constraints, synthesis implements a multibit component using multibit cells or single-bit cells.

When this option is set to the default of **false**, the **insert_dft** command treats the cells inside a multibit component as discrete sequential cells that can be reordered, split up, and rebalanced across scan chains as needed. Note that this does not affect the functional behavior of a multibit component as used by synthesis.

When this option is set to **true**, the **insert_dft** command treats each sequential multibit component as a scan segment. As a result, the cells inside a multibit component cannot be separated for length-balancing purposes. You can report the multibit scan segments that will be used by using the **preview_dft -show {segments}** command.

To report the multibit components identified by synthesis, use the **report_multibit** command.

-style multiplexed_flip_flop | clocked_scan | lssd | combinational | scan_enabled_lssd | none

Identifies the scan style. Select **none** if you have not selected a scan style for the design. By default, **insert_dft** uses the scan style value specified by environment variable **test_default_scan_style**.

-shared_scan_in pin_count

Specifies the number of scan-in pins to be shared between chains. The pins will be shared between chains which do not have a scan-in pin assigned to them. If there are existing scan-in pins in the design that are not already assigned to any chain, they will be used. If there are no existing pins that can be used, new pins are created and shared between the chains.

-exclude_elements exclude_list

Specifies the list or collection of design objects to exclude from scan chain insertion. You can specify leaf cells, hierarchical cells, or scan segment names. If hierarchical cells are specified, the specification applies to all sequential cells inside those cells.

Specification of scan segment names results in spurious UID-95 warnings, which can be ignored.

This option causes the specified cells to be excluded from scan stitching by the **insert_dft** command. It does not prevent the specified cells from being scan-replaced by the **compile -scan**, **compile_ultra -scan**, or **insert_dft** commands, and it does not cause excluded test-ready cells to be unscanned by the **insert_dft** command. To prevent cells from being scan-replaced, use the **set_scan_element false** command.

This option is cumulative with other previously specified **-exclude_elements** lists. Refer to the example at the end of this man page. Wildcards and collections are supported.

This option does not affect DFT connections to clock-gating cell test pins; use the **set_dft_clock_gating_configuration -exclude_elements** command instead.

-voltage_mixing true | false

When set to **true**, allows scan elements from different voltage domains into the same scan chain. This requires you to insert level shifters after scan insertion is complete. The default is **false**.

-power_domain_mixing true | false

When set to **true**, allows scan elements from different power domains into the same scan chain. The default is **false**.

-test_mode mode_name

Indicates which test mode the scan configuration applies to. The default is the last test mode specified with the **current_test_mode** command, or the last test mode created by **define_test_mode** if no current test mode has been set.

-domain_based_scan_enable true | false

When set to **true**, **insert_dft** creates one scan-enable signal per clock domain. This process is independent of the value of the **-clock_mixing** option. The default is **false**.

This option is intended to be used to create a core that can use pipelined scan-enable signals at a higher integration level.

To make domain-specific scan-enable connections for a subset of clocks instead of all clocks, use the **set_dft_signal -type ScanEnable -connect_to <clock>** command instead.

-reuse_mv_cells true | false

Specifies whether **insert_dft** should reuse existing level shifters, isolation cells, and enabled level shifters if they are on the scan path. When set to **true** (the default), **insert_dft** will attempt to reuse these cells to avoid creating dedicated test ports and additional multivoltage cells. When set to **false**, **insert_dft** will always create new multivoltage cells, creating new test ports as needed.

-pipeline_scan_enable true | false

When set to **true**, **insert_dft** creates a scan-enable pipelining stage for each scan-enable signal. The default is **false**.

-pipeline_fanout_limit *max_scan_cells*

When set to a positive integer value, **insert_dft** creates each scan-enable signal so that the number of flip-flops driven by the same scan-enable signal does not exceed the value. This option only takes effect if the **-pipeline_scan_enable** or **-domain_based_scan_enable** option (or both) is set to **true**. The default is to not apply a limit.

-create_test_clocks_by_system_clock_domain *true* | *false*

When **true**, **insert_dft** creates dedicated test clocks (for example, `test_scan_clock_a`) according to different system clocks. Thus, scan cells that have different system clocks have different test clocks. When **false** (the default), **insert_dft** creates test clocks without considering system clocks.

-replace *true* | *false*

When **true** (the default), **insert_dft** replaces sequential cells with scan cells, if the sequential cells are not violated by scan design rule checking. To disable scan replacement, set this option to **false**.

-hierarchical_isolation *true* | *false*

When **true**, **insert_dft** builds hierarchical isolation logic, so that dedicated subdesign scan-out signals are gated by the design scan-enable signal. This prevents long top-level scanout nets from toggling during functional operation. When **false** (the default), **insert_dft** does not add hierarchical isolation logic.

When this option is set to **true**, you cannot disable the use of shared subdesign scan-out signals by setting the **test_dedicated_subdesign_scan_outs** variable to **true**.

DESCRIPTION

There are four related options to control the number of scan chains. Only one option can take effect at a time. They are, in order of highest precedence first:

- **-max_length**
- **-chain_count** or **-count_per_domain**
- **-exact_length**

You can use these options to directly or indirectly specify the scan chain configuration for your design. If none of these options are specified, the tool builds the minimum number of scan chains consistent with the clock mixing constraints.

The following option controls clock domain identification:

-internal_clocks *single* | *none* | *multi*

Clocks driven by separate top-level input ports are always considered to be separate clocks for scan chain routing. The **-internal_clocks** option controls whether regions of the same clock signal, driven by multi-input combinational gates, should also be treated as separate clock domains. For designs where the entire clock tree is balanced through these multi-input combinational gates, you can specify a value of **none** to treat the entire clock port fanout as a single clock domain. For designs where clock latencies might differ in the fanout of each multi-input gate, it is recommended that you specify a value of **multi** to treat these regions as separate clock domains.

The following options control how separate clock domains can be mixed during scan chain routing:

-clock_mixing *no_mix* | *mix_edges* | *mix_clocks* | *mix_clocks_not_edges*
-mix_internal_clock_driver *true* | *false*

The **-clock_mixing option** allows you to control two independent aspects of mixing: whether separate clock domains can be mixed

together on the same scan chain, and whether scan cells clocked by different clock edge polarities can be mixed together on the same scan chain. Four possible values allow all four combinations of these two aspects to be specified. When separate clock domain mixing is disabled by specifying a **-clock_mixing** value of **no_mix** or **mix_edges**, the **-mix_internal_clock_driver** option can be used to allow only separate clock domains driven by the same input port (such as those created with the **-internal_clocks** option) to be mixed.

For all possible clock domain mixing configurations, lockup latches are inserted between the separate clock domains when they are mixed on the same scan chain.

EXAMPLES

The following example allows DFT Compiler to establish the maximum sequential length of scan chains (30 in this example):

```
prompt> set_scan_configuration -max_length 30
```

The following example demonstrates how a mix of object types can be provided to the **set_scan_configuration** command. In this example, the **-exclude_elements** option takes **s2/ff1** which is a cell name, **s1/f***, which is a wild card representation, **s0/1** which is a segment name, **\$foo** which is a collection, and **U5** which is a design instance name.

```
prompt> set foo [get_cell s4/*]
```

```
prompt> set_scan_config -exclude_elements \  
[list s2/ff1 s1/f* s0/1 $foo U5]
```

The following example allows DFT Compiler to mix scan cells in the same scan chain when a clock passes through multi-input combinational cells. Lockup latches will be inserted between scan cells driven by different multi-input combinational cells. Scan cells driven by separate clock input ports will not be mixed.

```
prompt> set_scan_configuration -internal_clocks multi
```

```
prompt> set_scan_configuration -clock_mixing no_mix
```

```
prompt> set_scan_configuration -mix_internal_clock_driver true
```

SEE ALSO

current_design(2)
insert_dft(2)
preview_dft(2)

set_scan_element

Sets the **scan_element** attribute on specified design objects, to determine whether scan replacement replaces them with scan cells.

SYNTAX

```
int set_scan_element  
    true | false  
    cell_design_ref_list
```

Data Types

cell_design_ref_list list

ARGUMENTS

true | false

The Boolean value with which to set the **scan_element** attribute on the *cell_design_ref_list*. When **true**, (the default), the specified objects are replaced by sequential cells in the design. When **false**, scan replacement is disabled.

cell_design_ref_list

A list of design objects for scan replacement. Objects must be of the following *sequential* types:

- cells (such as flip-flops and latches)
- hierarchical cells (containing flip-flops or latches)
- references
- library cells
- designs

Wildcards and collections are supported.

DESCRIPTION

Sets the **scan_element** attribute to **true** on objects in *cell_design_ref_list*, indicating that scan replacement is to replace them with equivalent scan cells and make them part of the scan path. This command affects scan replacement performed by the **insert_dft**, **compile -scan**, and **compile_ultra -scan** commands.

The default attribute value is **true**, which replaces all nonviolated sequential cells with equivalent scan cells for full scan designs. It is only necessary to explicitly set this value to **true** if you are altering the attribute value of a cell whose attribute was previously set to **false**.

Nonscan sequential cells determined to be violations by the **dft_drc** command are not scan-replaced, even if their **scan_element** attribute is set to **true**.

Sequential cells with the **scan_element** attribute set to **false** are not to be replaced by equivalent scan cells. If the attribute is applied before the first test-ready compile, the cells are never scan-replaced. If the attribute is set to **false** after a previous test-ready compile, the following behaviors apply:

- Subsequent test-ready compile commands will not unscan them.
- In wireload mode, the **insert_dft** command will unscan them (unless the **set_dft_insertion_configuration - synthesis_optimization** option is set to **none**).
- In topographical mode, the **insert_dft** command will keep the cells scan-replaced (to minimize layout disturbance) but will not include them on scan chains.

If the library has no nonscan cells, then scan cells with deasserted test pin connections are used. For hierarchical cells, the **scan_element** attribute value applies to all sequential leaf cells within the hierarchical cell. For references, library cells, or designs, the **scan_element** attribute value applies to the instances of these objects, or to the sequential leaf cells within the hierarchy of the instances. The effects of the **scan_element** attribute are applied hierarchically in a design. The value of the **scan_element** attribute on a lower-level object in the design takes precedence over the value of the **scan_element** attribute on a higher level object in the design. For example, if the value of the **scan_element** attribute on a higher-level object in the hierarchy is **false** and the value of the **scan_element** attribute on a sequential element inside the higher-level object is **true**, the sequential element is scan replaced. The **scan_element** attribute value on a cell instance takes precedence over the **scan_element** value on the reference for that cell. The **scan_element** attribute value on a library cell is treated as a technology limitation and cannot be overwritten in any way. The **set_scan_element** command supports full instance-based specification. However, the **scan_element** attributes on instances are specific to the current design. For example, if you start with a lower-level design, use the **set_scan_element** command to put **scan_element** attributes on instances in this context, and change the current design to a higher-level design, the **scan_element** attributes are no longer visible at the higher level. To remove individual **scan_element** attributes, use the **remove_attribute** command. To remove all attributes, use the **reset_design** command. To disable scan connections for CTL-modeled custom scan cells that do not use the synthesis-based scan replacement flow, use the **set_scan_configuration-exclude** command instead. This command does not affect DFT connections to clock-gating cell test pins; use the **set_dft_clock_gating_configuration-exclude_elements** command instead.

EXAMPLES

The following example specifies that the **insert_dft** command is not to scan replace three sequential cells in the current design.

```
prompt> set_scan_element false {U3_1 U3_2 U3_3}
```

The following example specifies that the **insert_dft** command is to scan replace every instance of cell DFF in the current design.

```
prompt> set_scan_element true [get_references DFF]
```

The following example specifies that the **insert_dft** command is not to scan replace instances of library cell DLATCH.

```
prompt> set_scan_element false tech_lib/DLATCH
```

The following example specifies that the **insert_dft** command is not to scan replace instances of design REGFILE (sequential cells within REGFILE are not to be replaced by equivalent scan cells).

```
prompt> set_scan_element false [get_designs REGFILE]
```

SEE ALSO

current_design(2)
dft_drc(2)
get_attribute(2)
insert_dft(2)
preview_dft(2)
remove_attribute(2)
reset_design(2)

set_scan_group

Specifies an unordered group of cells that are not yet connected, but should be kept together within a scan chain. Also identifies existing logic in the current design that is to be designated as a scan segment.

SYNTAX

```
status set_scan_group
  scan_group_name
  [-access signal_type_pin_pairs]
  [-include_elements object_list]
  [-serial_routed false | true]
  [-lockup_exists false | true]
  [-segment_length length_of_virtual_segment]
  [-clock top_level_clock_port]
  [-edge rising | falling]
  [-class occ | wrapper | input_wrapper | output_wrapper]
```

Data Types

<i>scan_group_name</i>	string

ARGUMENTS

scan_group_name

Specifies the name of the scan group. The scan group must be uniquely identified as a design object. The name of the scan group name must also be unique.

-access *signal_type_pin_pairs*

Lists ordered pairs, each consisting of a scan signal type and a design pin, indicating how the **insert_dft** command is to access the scan group. Valid signal types are ScanClock, ScanMasterClock, ScanSlaveClock, ScanEnable, ScanDataIn, and ScanDataOut. Validation ensures that specified signal types are consistent with the design scan style. Note that this list is mandatory when the **-serial_routed** option is set to **true**. This option is not supported when **-serial_routed** is set to **false**.

-include_elements *object_list*

Specifies a list of scan group components.

If the **-serial_routed** option is set to **false**, the list is unordered, and you can specify sequential cells, segment names, and design instances. Collections are supported.

If the **-serial_routed** option is set to **true**, the list is ordered, and you can specify only sequential cells. Collections are not supported for serially routed lists.

-serial_routed false | true

Identifies whether the scan group is composed of serially routed sequential cells or that you are specifying an unordered group of sequential cells. The default is **false**.

-lockup_exists false | true

Identifies whether the serially routed scan group has a lock-up latch at the end. The default is **false**.

This option can be used only when the **-serial_routed** option is set to **true**.

-segment_length length_of_virtual_segment

Specifies the length of the virtual scan segment. This option can be used only when **-serial_routed** is set to **true**. With this option, you must also use **-access** to specify the correct input and output access pins, and use **-clock** for top-level clock specifications with respect to this segment.

-clock top_level_clock_port

Specifies the top level clock port or pin associated with the virtual scan segment.

-edge rising | falling

Specifies the edge of the clock associated with the virtual scan segment. The scan architect uses this information for architecting the scan chains.

-class occ | wrapper | input_wrapper | output_wrapper

Specifies the class of the scan group.

The **occ** keyword specifies that the scan group describes an on-chip clocking (OCC) clock chain segment. Scan groups of **-class occ** are used only during OCC flows.

The **wrapper** keyword specifies that the scan group describes a wrapper chain segment. Scan groups of **-class wrapper** are used only during core wrapping flows.

The **input_wrapper** keyword specifies that the scan group describes an input wrapper chain segment. Scan groups of **-class input_wrapper** are used only during core wrapping flows.

The **output_wrapper** keyword specifies that the scan group describes an output wrapper chain segment. Scan groups of **-class output_wrapper** are used only during core wrapping flows.

DESCRIPTION

The **set_scan_group** command identifies existing logic in the current design that is to be designated as a scan group. Scan groups implement scan in a particular scan style. DFT Compiler can include scan groups in scan chains by connecting their access pins (when serially routed), or maintain the cells' grouping without assuming they are serially connected to each other. The **insert_dft** command does not scan-replace scan group members explicitly.

When the **-serial_routed** option is set to **true**, the **set_scan_group** command allocates sequential cells to scan groups and identifies design pins as scan group access pins with particular scan signal-type semantics. The **set_scan_configuration** command used with the **-style** option determines the scan group scan styles.

The **set_scan_group** command options are not incremental. A **set_scan_group** option that specifies a scan group name overwrites any previous **set_scan_group** command specification of the same name and scan style.

The **set_scan_group** command accepts a list of members. By default, the list of members is unordered and not serially connected. This list must be ordered when **-serial_routed** is set to **true**.

The scan group members should belong to the same clock domain and same edge. If the scan group is provided with elements from a different clock domain, an error message is displayed during scan architecting and the scan group specification is discarded.

The **set_scan_group** command does not accept collections as input.

The **set_scan_group** command accepts design instances as scan group members. It adds every scannable cell in the design instance to the scan group at the specified position, in alphanumeric order. This specification is valid only when **-serial_routed** is set to **false**.

The **-segment_length** option specifies the length of the virtual scan segment. Virtual segments are segments for which you specify the scan segment input and the scan segment output using the **-access** option, and the scan segment clock association using **-clock** option. If you do not specify the **-include_elements** option but you want specify the length of a segment, use the **-segment_length** option. The tool connects only to the hookup points during scan chain construction and does not trace within the hookup points (or virtual scan segment). The segment length you provide is considered for top-level scan chain balancing.

Note that the post-DFT DRC and SCANDEF generation support is not available when virtual scan segments are present in the scan chain.

The **-clock** option specifies the top-level clock that drives the virtual scan segment. If you specify the **-segment_length** option, you must also specify a valid **-clock** specification.

Ensure that the clock name specified as part of the **-clock** option is already defined using the **set_dft_signal** command. Alternatively, run the **set_scan_group** command after creating the test protocol and inferring the clocks and asynchronous signals.

The **-edge** option specifies the controlling edge of the clock with respect to the virtual scan segment. The controlling edge is specified as **rising** or **falling**. The default is **rising**.

When the **-access** option is used, the access pin signal types must be consistent with the segment scan style. Access pin directions must be consistent with their signal types. Access pins cannot be associated with more than one signal type. Scan groups cannot have more than one ScanDataIn or ScanDataOut access pin. Scan groups cannot contain duplicate members; sequential cells cannot belong to more than one scan group. Scan group members cannot also belong to scan chains. In other words, a sequential cell element that belongs to a scan group cannot be specified as part of the **set_scan_path** command.

A scan group does not accept another scan group as a member; nested scan groups are not supported.

Valid scan group error messages are TEST-400 and TEST-700. TEST-400 is displayed when scan group elements belong to more than one clock domain or have the same clock domain but a different edge. TEST-700 is displayed when scan group elements have a nonexistent element specified as part of the scan group specification.

Use the **preview_dft** command with the **-bsd** option set to all to review your scan group specifications. Use the **remove_scan_group** command with *scan_group_name* argument to delete an existing scan group specification.

EXAMPLES

The following example identifies an embedded shift register in the multiplexed flip-flop scan style. This specification can be completed in a single **set_scan_group** command by including the identification of the ScanEnable access pin in the first **-access** option specification.

```
prompt> set_scan_group my_shift_reg \
    -access [list ScanDataIn P/B/U7/si ScanDataOut P/B/U9/QN] \
    -include_elements [list P/B/U7 P/B/U8 P/B/U9] \
    -serial_routed true
```

```
prompt> set_scan_group my_unconnected_group \
    -include_elements [list U1 U2]
```

The following is an example of a virtual scan segment controlled by the rising edge of the clock tck:

```
prompt> set_scan_group virtual_scan_segment \  
-access [list ScanDataIn P/B/U7/si ScanDataOut P/B/U9/QN] \  
-segment_length 20 \  
-clock tck \  
-edge rising \  
-serial_routed true
```

The following is an example of a virtual scan segment controlled by the falling edge of the clock tck:

```
prompt> set_scan_group virtual_scan_segment \  
-access [list ScanDataIn P/B/U7/si ScanDataOut P/B/U9/QN] \  
-segment_length 20 \  
-clock tck \  
-edge falling \  
-serial_routed true
```

SEE ALSO

insert_dft(2)
preview_dft(2)
remove_scan_group(2)

set_scan_link

Declares a scan link for the current design.

SYNTAX

```
int set_scan_link  
    scan_link_name Wire | Lockup  
    [-test_mode mode_name]
```

Data Types

scan_link_name string

ARGUMENTS

scan_link_name

Specifies the name of the scan link. Substitute the name you want for *scan_link_name*.

-test_mode *mode_name*

Specifies the test mode for the specification. The default is current mode.

DESCRIPTION

Declares a scan link for the current design. Scan links connect scan cells, scan segments, and scan ports within scan chains. DFT Compiler supports scan links that are implemented as wires (type Wire) and scan-out lock-up latches (type Lockup).

This command reserves words for particular types of scan links. Use these words within **set_scan_path** commands to specify scan links of various types. You can use scan links to insert lock-up latches at specific scan chain locations and correct timing problems. You can also use wire scan links to override the automated insertion of lock-up latches. The **scan_link_name** option must uniquely identify the scan link as a design object.

You can use the **-test_mode** option to specify the test mode.

To review your scan link specifications, use the **preview_dft** command with the **-show** option set to **cells**. To create scan links and add them to the current design, use the **insert_dft** command. To delete scan link specifications, use the **remove_scan_link** command.

The **set_scan_link** command does not invalidate **dft_drc** modeling information.

EXAMPLES

The following example declares a lock-up latch named `a_lckup` in `Internal_scan` mode and uses it to resynchronize the scan outputs of the elements named `A` and `F` of chain `chain3`.

```
prompt> current_design WC66  
prompt> set_scan_link a_lckup Lockup -test_mode Internal_scan  
prompt> set_scan_path chain3 {A a_lckup B C D E F a_lckup G}  
prompt> insert_dft
```

SEE ALSO

- `current_design(2)`
- `dft_drc(2)`
- `insert_dft(2)`
- `preview_dft(2)`
- `remove_scan_link(2)`
- `report_scan_link(2)`
- `set_dft_signal(2)`
- `set_dont_touch(2)`
- `set_scan_configuration(2)`
- `set_scan_element(2)`
- `set_scan_path(2)`
- `write(2)`

set_scan_path

Specifies a scan chain for the current design.

SYNTAX

```
integer set_scan_path
    scan_chain_name
    [-ordered_elements ordered_list]
    [-head_elements head_list]
    [-tail_elements tail_list]
    [-include_elements include_list]
    [-infer_dft_signals]
    [-dedicated_scan_out true | false]
    [-complete true | false]
    [-exact_length length]
    [-bsd_style global | synchronous | asynchronous]
    [-insert_terminal_lockup true | false]
    [-scan_master_clock clock_name]
    [-edge rising | falling]
    [-scan_slave_clock clock_name]
    [-scan_enable port_name]
    [-scan_data_in port_name]
    [-scan_data_out port_name]
    [-test_mode mode_name]
    [-view view_name]
    [-class scan | wrapper | bsd | occ | spc | ieee_1500 | ieee_1500_ring | pco_wrapper]
    [-opcode opcode_value]
    [-init_data init_value]
    [-hookup hookup_pin_list]
    [-input_wrapper_cells_only enable | disable]
    [-output_wrapper_cells_only enable | disable]
    [-pipeline_head_registers scan_chain_element_names]
    [-pipeline_tail_registers scan_chain_element_names]
    [-sel_wir_position SelectWIR_position_in_ring_data_register_:_closest_to_WSI/TDI_or_WSO/TDO]
```

Data Types

<i>scan_chain_name</i>	string
<i>ordered_list</i>	list
<i>head_list</i>	list
<i>tail_list</i>	list
<i>include_list</i>	list
<i>length</i>	integer
<i>clock_name</i>	string
<i>port_name</i>	string
<i>mode_name</i>	string
<i>view_name</i>	string

```

opcode_value      string
init_value        string
hookup_pin_list   list
scan_chain_element_names  list

```

ARGUMENTS

scan_chain_name

Specifies a name for the scan chain being specified.

-ordered_elements *ordered_list*

Specifies an ordered list of scan chain elements to be included in the scan chain. This list can include names of sequential cells, design instances, scan segments, and scan links. The input is accepted as a simple list. Wildcards are supported, but collections are not.

-head_elements *head_list*

Specifies an ordered list of scan chain elements to be placed at the beginning of the scan chain. This list can include names of sequential cells, design instances, scan segments, and scan links. The input is accepted as a simple list. Wildcards are supported, but collections are not.

-tail_elements *tail_list*

Specifies an ordered list of scan chain elements to be placed at the end of the scan chain. This list can include names of sequential cells, design instances, scan segments, and scan links. The input is accepted as a simple list. Wildcards are supported, but collections are not.

-include_elements *include_list*

Specifies a list of scan chain elements to be included in the scan chain. Unlike an ordered list, the scan architect is free to place them anywhere in the scan chain. This list can include names of sequential cells, design instances, scan segments, and scan links. The input is accepted as a simple list. Wildcards are supported, but collections are not.

-infer_dft_signals

Specifies that the **set_scan_path** command automatically infers the ScanDataIn, ScanDataOut, and ScanEnable DFT signals that are normally specified using the **set_dft_signal** command. This option is only applicable when you are using this command in a scan extraction flow.

Note that if you use this option, and you later use the **remove_scan_path** command to remove scan paths, the signals that were inferred are not removed. You must use the **remove_dft_signal** command to explicitly remove these DFT signals.

-dedicated_scan_out true | false

Forces the use of a dedicated scan-out port, which allows the **insert_dft** command to reuse a functional output port driven by this scan chain as a scan-out port. When this option is set to **true**, the **set_scan_path** command always uses a dedicated scan-out port for this scan chain.

-complete true | false

Indicates whether **insert_dft** can add components to a specified scan chain. When **true**, **insert_dft** does not add any other scan cells to this scan chain and treats it as a complete scan chain. When **false** (the default), **insert_dft** can add more scan cells to this specification to balance scan chains. The new scan cells are embedded at the beginning of the scan chain (after any **head_elements** scan cells).

-exact_length *length*

Specifies that **insert_dft** command should build the *scan_chain_name* scan chain with the exact sequential length indicated by *length*. The value must be a positive integer. If the specification cannot be met due to other scan requirements, the tool generates a warning stating that the specified scan chain requirements cannot be honored, and then it proceeds with normal scan architecting.

If you specify **-class bsd**, this option allows you to specify the length of the boundary-scan test data register being specified.

This option is not used when describing the length of an existing segment by specifying the list of elements.

-bsd_style *global* | *synchronous* | *asynchronous*

Specifies the boundary-scan style for a user-defined test data register (UTDR) definition. The default is **global**, which uses the global style specified by the **-style** option of the **set_bsd_configuration** command.

-insert_terminal_lockup *true* | *false*

Specifies a lock-up latch to be associated at the end of the scan chain. Scan architect takes the user-defined lock-up latch specification into consideration while building scan chains.

-scan_master_clock *clock_name*

Specifies a clock to be associated with a scan chain. The scan architect takes the user-defined clock domain into consideration while building scan chains. The clock edge is determined by the **-edge** option, which defaults to rising-edge scan cells.

If the same clock edge specification applies to more than one scan chain specification, the scan architect attempts to balance the clock domain elements across the scan chains.

-edge *rising* | *falling*

Specifies the edge of the *scan_master_clock* associated with the scan chain. The scan architect includes only the elements controlled by the specified edge of the *scan_master_clock* in the scan chain. The default is **rising**.

-scan_slave_clock *clock_name*

Specifies a clock to be associated with a scan chain. The scan architect takes the user-defined clock domain into consideration while building scan chains.

-scan_enable *port_name*

Specifies a scan enable to be associated with a scan chain. The scan architect takes the specified scan-enable port into consideration while allocating a scan-enable signal to the scan chain. The specified scan-enable port is dedicated to the scan chain; it will not be used to drive scan enable pins of flip-flops in other chains. If the internal pins flow is enabled using the **set_dft_drc_configuration** command, a hookup pin can be specified as a scan-enable signal for a scan chain.

When you specify **-class wrapper**, the specified port name can be a wrapper shift enable signal.

-scan_data_in *port_name*

Specifies a scan-in port to be associated with a scan chain. The scan architect takes the user-defined scan-in port into consideration while building scan chains. If the internal pins flow is enabled using the **set_dft_drc_configuration** command, a hookup pin can be specified as a scan-in signal for a scan chain.

-scan_data_out *port_name*

Specifies a scan-out port to be associated with a scan chain. The scan architect takes the user-defined scan-out port into consideration while building scan chains. If the internal pins flow is enabled using the **set_dft_drc_configuration** command, a hookup pin can be specified as a scan-out signal for a scan chain.

-test_mode *mode_name*

Indicates the test mode to which the scan path constraint applies. The default is the current test mode, if user-defined test modes have been created, or the `Internal_scan` mode if no test modes have been created.

Note that unlike most commands, the **set_scan_path** command does not apply a global specification to all test modes when the **-test_mode** option is omitted and no current test mode has been defined. To apply a scan path specification to all test modes, you must explicitly specify the **-test_mode all** option.

-view view_name

Indicates the view to which the specification applies. The following views are valid:

- **existing_dft** indicates that the specification refers to the existing usage of a scan chain. Use this when working with DFT-inserted designs. For example, to describe to the tool that chain A is being used, enter the following:

```
set_scan_path C1 -view existing_dft -ordered_elements {reg0 reg1 reg2}
```

- **spec** (default) indicates that the specification refers to scan chains that the tool will create during DFT insertion. This is used when preparing a design for DFT insertion. For example, to prescribe to the tool that a scan chain A must be created in a certain way, you would do the following

```
set_scan_path C1 -view spec -ordered_elements {reg0 reg1 reg2}
```

-class scan | wrapper | bsd | occ | spc | ieee_1500 | ieee_1500_ring | pco_wrapper

Specifies the class of the scan chain. The allowed values are:

- **scan** (default) specifies that the chain is a scan chain. Chains of class **scan** are used only during scan insertion.
- **wrapper** specifies that the chain is a wrapper chain of a wrapped core. Chains of class **wrapper** are used only during core wrapping insertion.
- **bsd** specifies that the chain is a BSR chain or a BSD register. Chains and registers of class **bsd** are used only during BSD insertion.
- **occ** specifies that the chain is a clock chain for an on-chip clocking (OCC) controller. Chains of class **occ** are used only in an OCC flow.
- **spc** specifies that the chain is a shift power control (SPC) chain, used by the shift power groups feature. Chains of class **spc** are used only when the SPC feature is enabled.
- **ieee_1500** specifies that the chain is an IEEE 1500 register or is accessed by the IEEE 1500 controller. Chains and registers of class **ieee_1500** are used only when describing IEEE 1500 logic.
- **ieee_1500_ring** specifies that the chain describes an IEEE 1500 ring. Chains of class **ieee_1500_ring** are used only when describing IEEE 1500 logic.
- **pco_wrapper** specifies that the chain is to be the control chain in a power controller override (PCO) flow. Chains of class **pco_wrapper** must be defined as **-view spec**. The specifications can include scan-in and scan-out ports but cannot include any scan cell elements.

Scan chains defined as a certain class automatically include elements of that class, unless limited or explicitly specified by other options of this command. The exceptions are the **occ** and **spc** classes, which require the scan elements to be explicitly specified if cores containing **occ** or **spc** chains are present.

-opcode opcode_value

Specifies the value that, when loaded into the IEEE 1500 wrapper instruction register (WIR), selects the data register being defined by the current command. The *opcode_value* is a binary string.

The default is to choose any suitable available encoding. This option is only valid for IEEE 1500 data register definitions. It requires that the **-class ieee_1500** option also be used.

-init_data *init_value*

Specifies a user-defined test data register (UTDR) initialization value associated with the current **set_scan_path** specification. The *init_value* argument is a binary string.

With **-class *bsd***, the option specifies the value to be loaded into the UTDR described by the current **set_scan_path** command when an IEEE 1149.1 instruction is loaded for test-mode control.

With **-class *ieee_1500_ring***, the option specifies the value to be loaded into the ring selection UTDR to select the ring described by the current **set_scan_path** command.

Other **-class** values are not supported.

This option is used only by the SHS and DFTMAX interoperability flow.

-hookup *hookup_pin_list*

Specifies the hookup pin names associated with the scan chain. The *hookup_pin_list* can contain hierarchical pin names or pin objects.

These pin names are used in defining a boundary-scan register (of class **bsd**). The signal types for these pins are specified using *set_dft_signal* command. This list of pins, along with the *set_dft_signal* signal definitions, defines the hookups needed for the boundary-scan register.

By default, capture and update clock pins of the boundary-scan test data register are not gated, which means the pins are always driven by TCK. This behavior is controlled by the **test_bsd_synthesis_gated_tck** variable. For more information, see the man page.

There is no default for this option.

-input_wrapper_cells_only *enable* | *disable*

Specifies whether or not only input wrapper cells should be used in the chain. This option is specific to chains of class **wrapper**.

The default for this option is **disable**.

-output_wrapper_cells_only *enable* | *disable*

Specifies whether or not only output wrapper cells should be used in the chain. This option is specific to chains of class **wrapper**.

The default for this option is **disable**.

-pipeline_head_registers *scan_chain_element_names*

Specifies an ordered list of existing head pipelined scan data registers, starting from the scan-in port. In regular scan mode, the scan data goes through these elements after the scan-in port and before the first scan cell of the specified scan chain. In compressed scan mode, the scan data goes through these elements after the scan-in port and before the adaptive scan decompressor. These elements are automatically excluded from scan architecting. This ordered list must only include instance names of sequential cells. If the scan-in port attached to the scan chain does not have a hookup pin specified, the last element that is specified drives the scan chain using the first output pin found for that cell. The input is accepted as a simple list. Wildcards and collections are not supported.

-pipeline_tail_registers *scan_chain_element_names*

Specifies an ordered list of existing tail pipelined scan data registers, ending at the scan-out port. In regular mode scan, the scan data goes through these elements after the last scan cell of the specified scan chain and before the scan output port. In compressed scan mode, the scan data goes through these elements after the adaptive scan compressor output and before the scan output port. These elements are automatically excluded from scan architecting. This ordered list must only include instance names of sequential cells. If the scan output port attached to the scan chain does not have a hookup pin specified, the first element that is listed observes the scan chain using the first usable input pin found for that cell. The input is accepted as a simple list. Wildcards and collections are not supported.

DESCRIPTION

This command specifies a scan chain for the current design. The command allocates scan cells, scan segments, and scan links to scan chains, and specifies scan chain orderings. The **set_scan_configuration** command **-style** option determines the chain scan style. It can be used either to describe the existing usage or to prescribe the usage during implementation.

Scan chain elements cannot belong to more than one chain. When **set_scan_path** options conflict, the most recent command overrides the previous command.

The **set_scan_path** options are not incremental. A **set_scan_path** option specified for a given chain name overwrites any previous **set_scan_path** option previously applied to that same chain name.

The **set_scan_path** command accepts design instances as scan chain elements. When specified, they include every scannable cell in the design instance.

Scan segments can be subdesign scan chains. Suppose the hierarchical design instance *instance_name* has a scan chain of **report_dft**. The **-scan_path** option reports as "Complete scan chain #n." You can include this subdesign scan chain in a scan chain by referring to it as **instance_name/n**.

Scan chain orderings must satisfy clock domain constraints asserted by using the **set_scan_configuration** command with the **-clock_mixing** option. The **insert_dft** command resynchronizes nonfunctional scan chains by using lockup latches. DFT Compiler reorders nonfunctional scan chains if you disable lockup latch insertion by using the **set_scan_configuration** command with the **-add_lockup** option.

The **set_scan_path** command does not invalidate **dft_drc** modeling information.

Use the **report_scan_path** command to review your scan chain specification. Use the **insert_dft** command to implement the scan chain. Use the **remove_scan_path** command to delete scan chain specifications.

In the case of multimode scan insertion, if you have previously defined several test modes using the **create_test_schedule** or **define_test_mode** command, the default **set_scan_path** command is not associated with any mode and is not honored. Use the **-test_mode** option to specify the test mode to which you want your scan path constraint to apply.

When using this command in a scan extraction flow, either specify the existing DFT signals using the **set_dft_signal -view existing** command prior to running the **set_scan_path** command, or use **set_scan_path** with the **-infer_dft_signals** option to infer these signals.

To define an existing IEEE 1500 register, specify **-class ieee_1500** and specify the ordered list of register cells with the **-ordered_elements** option. The following scan path names are special values that represent part of the IEEE 1500 architecture:

- **WIR** - wrapper instruction register
- **WBY** - wrapper bypass register

Other scan path names represent user-defined core data registers (CDRs). A user-defined CDR serves as the test-mode control data register (TMCDR) unless the WIR is used directly for test-mode control. For non-WIR register definitions, you can define the WIR opcode that enables the register by using the **-opcode** option.

In a boundary-scan flow, to define a user-defined test data register (UTDR) accessed by the TAP controller, specify **-class bsd** and specify the register access pins using **-hookup**. You must also specify the signal types of the hookup pins by using the **set_dft_signal** command.

EXAMPLES

The following example defines a complete scan chain named C1, consisting of sequential cells A, B, and C in reverse alphanumeric order.

```
prompt> current_design WC66
prompt> set_scan_path C1 -include_elements {C B A} -complete true
```

The following example shows how you can use wildcards to specify design objects by converting a collection to a name list:

```
prompt> set_scan_path C1 -include_elements \
[get_object_names [get_cells {*cache*reg*}]]
```

The following example shows how to use the **-head_elements**, **-tail_elements**, and **-include_elements** options to specify that the elements from the U4 instance are at the beginning of the chain and the elements from the U5 instance are at the end of the chain. The elements from U3 are ordered and scan architect does not change the order of the elements in this instance during architecting. The **-include_elements** option ensures that the U2/2 segment is included somewhere in scan chain 1.

```
prompt> set_scan_path C1 -view spec -head_elements {U4} \
-include_elements {U2/2} -ordered_elements {U3} \
-tail_elements {U5}
```

The following example shows how to build scan chains with user-defined sequential lengths and/or clocks in a design with 25 scan cells, 10 of which belong to clock domain A and 15 of which belong to clock domain B. This builds C1 with 7 sequential elements clocked with A, C2 with 15 sequential elements clocked with B and a tool-named chain with the 3 remaining elements clocked with A.

```
prompt> set_scan_configuration -clock_mixing no_mix
prompt> set_scan_path C1 -exact_length 7 -clock A
prompt> set_scan_path C2 -scan_master_clock B
```

The following examples show how to use the **-edge** option along with the **-scan_master_clock** option. First, the `\fbset_scan_path` command assigns all elements controlled by the rising edge of the clk1 clock to C1, and then assigns all elements controlled by the falling edge of the clk1 clock to C2.

Consider **set_scan_path** specifications for C3 and C4 below. If there are 10 elements controlled by the rising edge of clock clk2, then C3 and C4 will get 5 elements each.

```
prompt> set_scan_path C1 -view spec -scan_master_clock clk1 -edge rising
prompt> set_scan_path C2 -view spec -scan_master_clock clk1 -edge falling

prompt> set_scan_path C3 -view spec -scan_master_clock clk2 -edge rising
prompt> set_scan_path C4 -view spec -scan_master_clock clk2 -edge rising
```

The following example shows how to specify the DFT signals and scan path for a design with existing scan chains:

```
prompt> set_dft_signal -view existing_dft \
-type ScanDataIn -port test_si
prompt> set_dft_signal -view existing_dft \
-type ScanDataOut -port test_so
prompt> set_dft_signal -view existing_dft \
-type ScanEnable -port test_se
prompt> set_dft_signal -view existing_dft \
-type MasterClock -port test_clk -timing {45 55}
prompt> set_dft_signal -view existing_dft \
```

```
-type ScanMasterClock -port test_clk -timing {45 55}
prompt> set_scan_path C1 -view existing_dft \
  -scan_data_in test_si -scan_data_out test_so -scan_enable test_se
```

The following example shows how to specify a scan path such that the first cell of the scan chain is connected to the output of pipeline register head_reg_2 and the last cell of the scan chain is connected to the input of pipeline register tail_reg.

```
prompt> set_scan_path C1 -view spec \
  -pipeline_head_registers {head_reg_1 head_reg_2} \
  -pipeline_tail_registers {tail_reg}
```

The following example shows how to specify a scan path for a design with existing scan chains, using the **set_scan_path -infer_dft_signals** option to infer the existing DFT signals:

```
prompt> set_scan_path C1 -view existing_dft -infer_dft_signals
```

The following example shows how to specify a boundary-scan register (BSR) chain for use with boundary-scan insertion. In this example, in1, in2, en, out1, out2 are ports of the current design.

```
prompt> set_scan_path -class bsd C_BSR -view spec \
  -ordered_elements {in1 in2 en out1 out2}
```

The following example shows how to specify a short BSR chain for use with boundary-scan synthesis. In this example, in1, in2 are the ports of the current design.

```
prompt> set_scan_path -class bsd C_SHORT_BSR -view spec \
  -ordered_elements {in1 in2}
```

The following example defines a set of existing, preconnected IEEE 1500 registers in the design:

```
prompt> set_scan_path WIR -class ieee_1500 -view existing_dft -test_mode all \
  -ordered_elements {WIR_reg[1] WIR_reg[0]}
prompt> set_scan_path WDR -class ieee_1500 -view existing_dft -test_mode all \
  -opcode 000001 -ordered_elements {TMCDR_reg[1] TMCDR_reg[0]}
prompt> define_test_mode WS_BYPASS -usage wrp_bypass \
  -encoding {WIR_reg[1] 1 WIR_reg[0] 1}
prompt> set_scan_path WBY -class ieee_1500 -view existing_dft -test_mode WS_BYPASS \
  -ordered_elements {WBY_reg}
```

The following example shows how to specify a user-defined test data register (UTDR) for use with boundary-scan synthesis. Here, I1/si, I1/so, I1/se, I1/clock, I1/inst_en are active-high access pins of the UTDR that are to be hooked up to boundary-scan logic during boundary-scan insertion. Pin I1/reset is an active-low access pin; note that it is excluded from the **set_scan_path** hookup pin list.

```
prompt> set_dft_signal -type tdi -view spec -hookup_pin I1/si
prompt> set_dft_signal -type tdo -view spec -hookup_pin I1/so
prompt> set_dft_signal -type bsd_shift_en -view spec -hookup_pin I1/se
prompt> set_dft_signal -type capture_clk -view spec -hookup_pin I1/clock
prompt> set_dft_signal -type inst_enable -view spec -hookup_pin I1/inst_en
prompt> set_dft_signal -type bsd_reset -view spec -hookup_pin I1/reset \
  -active_state 0
prompt> set_scan_path -class bsd MY_USER_REG -view spec -exact_length 10 \
  -hookup {I1/si I1/so I1/se I1/clock I1/inst_en}
```

SEE ALSO

current_design(2)

```
define_test_mode(2)
dft_drc(2)
insert_dft(2)
preview_dft(2)
remove_scan_path(2)
report_scan_path(2)
set_dft_signal(2)
set_dont_touch(2)
set_scan_compression_configuration(2)
set_scan_configuration(2)
set_scan_element(2)
set_scan_link(2)
write(2)
```

set_scan_register_type

Specifies a list of scan sequential cells from the target library that are to be used by **insert_dft** or **compile -scan** when scan replacing designs or cell instances.

SYNTAX

```
int set_scan_register_type  
  [-exact]  
  [-type example_scan_seq_cell_list]  
  [cell_or_design_list]
```

Data Types

```
example_scan_seq_cell_list  list  
cell_or_design_list        list
```

ARGUMENTS

-exact

If specified, indicates that the *example_scan_seq_cell_list* is to be honored even during backend delay and area optimization done by **insert_dft** or by subsequent synthesis operations (for example, **compile -incremental**). If **-exact** is not specified, backend optimization can optimize scan cell instances to use scan cells that are not in the *example_scan_seq_cell_list*.

-type *example_scan_seq_cell_list*

Specifies a list of scan sequential cells from the target library, to be used by **insert_dft** or **compile -scan** as the scan equivalent for the current design or for sequential cells in the *cell_or_design_list* during scan replacement.

cell_or_design_list

Specifies a list of sequential nonscan cells or designs that are to be replaced by the scan cells in *example_scan_seq_cell_list* during scan replacement. The default is the current design.

DESCRIPTION

The **set_scan_register_type** command specifies a list of scan sequential cells from the target library to be used by **insert_dft** or **compile -scan** as the scan equivalents for specified nonscan sequential cells during scan replacement.

To determine the current settings resulting from previous use of the **set_scan_register_type** command, execute **report_test -register**. To remove **set_scan_register_type** settings currently in effect, execute **remove_scan_register_type**.

Specifying **-exact** indicates that only cells from the **-type** list can be used as a scan replacements for those cells, even during scan optimization.

To effectively use the **set_scan_register_type** command, it is important to understand the scan insertion process. The process of mapping sequential cells into scan flip-flops and latches consists of three steps:

1. The **compile -scan** command maps each sequential cell in the generic design description into an initial nonscan latch or flip-flop from the target library. Non-scan sequential cells are mapped to an initial latch or flip-flop from the target library. In the absence of any **set_scan_register_type** specification, the smallest area-cost flip-flop or latch is chosen. For a design or cell instance that has a **set_scan_register_type** setting in effect, the nonscan equivalent of a scan cell in the *example_scan_seq_cell_list* is chosen.
2. The **compile -scan** or **insert_dft** command replaces the nonscan sequential cells with scan sequential cells, using only the scan cells specified by the **set_scan_register_type** command, where applicable. If **compile -scan** or **insert_dft** is unable to use a scan cell from the *example_scan_seq_cell_list*, it uses the best possible matching scan cell from the target library and issues a warning.
3. If non mapping was possible with fun-id. The scan replacement table specified using **set_scan_replacement** is searched for a one-to-one mapping for the design's scan style.
4. If the **-exact** option is not used in the **set_scan_register_type** command, **compile -scan** or **insert_dft** attempts to remap each scan sequential cell into another component from the target library to optimize the delay or area characteristics of the circuit. If the **-exact** option is used, optimization is restricted to using the scan cells in the *example_scan_seq_cell_list*.

EXAMPLES

In the following example, the scan register type for the current design is to be SDFLOP in the target library. (The DFLOP is a non-scan equivalent of this SDFLOP.) During scan replacement, the scan sequential cell used is SDFLOP, but during scan cell optimization, other scan cells from the target library could be used.

```
prompt> set_scan_register_type -type SDFLOP
```

In the following example, the scan register type for the current design is to be SDFLOP, and this cell is to be used as an exact scan replacement. During scan replacement, the scan sequential cell used is SDFLOP, and remains so during scan cell optimization.

```
prompt> set_scan_register_type -exact -type SDFLOP
```

The following example indicates that the scan register type for the current design is to be from the list of cells SD1FLOP SD2FLOP. During scan replacement, the scan sequential cell used is either an SD1FLOP or an SD2FLOP.

```
prompt> set_scan_register_type -type {SD1FLOP SD2FLOP}
```

SEE ALSO

compile(2)
current_design(2)
insert_dft(2)
set_scan_replacement(2)
remove_scan_register_type(2)
reset_design(2)
target_library(3)

set_scan_replacement

Specifies a table of one-to-one mappings of flip-flops to their equivalent scan flip-flops from the target library that are to be used by **insert_dft** when scan replacing cell instances.

Note that as of 2006.06-SP1, **compile -scan** and **compile_ultra -scan** map to scan cells directly, rather than doing scan replacement. So, this command has no effect on **compile -scan** and **compile_ultra -scan**.

SYNTAX

```
status set_scan_replacement
  [-nonscan non_scan_seq_cell_list]
  [-lssd lssd_rep_cell]
  [-multiplexed_flip_flop muxed_scan_cell]
  [-clocked_scan clocked_scan_cell]
  [-combinational combinational_scan_cell]
  [-scan_enabled_lssd scan_enabled_lssd_cell]
```

Data Types

```
non_scan_seq_cell_list  list
lssd_rep_cell           string
muxed_scan_cell        string
clocked_scan_cell      string
combinational_scan_cell string
scan_enabled_lssd_cell string
```

ARGUMENTS

-nonscan *non_scan_seq_cell_list*

Specifies the set of cells for which a scan replacement is specified using the **set_scan_replacement** command.

-lssd *lssd_rep_cell*

Specifies the scan replacement flip-flop for all flip-flops specified in the **-nonscan** option in the LSSD scan methodology.

-multiplexed_flip_flop *muxed_scan_cell*

Specifies the scan replacement flip-flop for all flip-flops specified in the **-nonscan** option in the multiplexed flip-flop scan methodology.

-clocked_scan *clocked_scan_cell*

Specifies the scan replacement flip-flop for all flip-flops specified in the **-nonscan** option in the clocked-scan methodology.

-combinational *combinational_scan_cell*

Specifies the scan replacement flip-flop for all flip-flops specified in the **-nonscan** option in the combinational scan methodology.

-scan_enabled_lssd scan_enabled_lssd_cell

Specifies the scan replacement flip-flop for all flip-flops specified in the **-nonscan** option in the scan-enabled LSSD scan methodology.

DESCRIPTION

The **set_scan_replacement** command indicates that the instances of cells in *non_scan_seq_cell_list* for the **-nonscan** option will be replaced by the corresponding scan cell specified in each of the scan style options.

The flip-flops specified in all options except **-nonscan** must be valid scan cells for the corresponding scan methodology.

The **set_scan_replacement** command can be used to specify a one-to-one mapping of nonscan flip-flops to corresponding scan flip-flops for each scan methodology.

The **insert_dft** command uses this mapping table to do scan replacement. If the mapping is not a valid scan cell for the corresponding scan methodology, **insert_dft** will reject the mapping and use other mapping methods, such as function ID or Boolean matching. Boolean matching can be time consuming, but ensures that a scan flip-flop that satisfies the design constraints is selected.

This mapping does not ensure that the design constraints are met before mapping. This command should be used only when you are sure that your constraints will be met by this mapping.

EXAMPLES

In the following example, all instances of FD1 will be replaced by the scan flip-flop FD1S in the multiplexed scan style:

```
prompt> set_scan_replacement -nonscan FD1 -multiplexed_flop_flop FD1S
```

In the following example, the instances of FD1, FD2 and FD3 will be replaced by FDLS in the LSSD scan methodology:

```
prompt> set_scan_replacement -nonscan {FD1 FD2 FD3} -lssd FDLS
```

SEE ALSO

current_design(2)
insert_dft(2)
remove_scan_register_type(2)
remove_scan_replacement(2)
report_scan_replacement(2)
reset_design(2)
translate(2)
target_library(3)

set_scan_skew_group

Defines a scan skew group of scan cells, which might have a different clock latency characteristic than other parts of the design.

SYNTAX

```
status set_scan_skew_group  
  scan_skew_group_name  
  -include_elements object_list
```

Data Types

```
scan_skew_group_name  string  
object_list           list
```

ARGUMENTS

scan_skew_group_name

Specifies the name of the scan skew group to define.

A scan skew group is referenced by name. You can use any name, as long as it has not previously been used to define another scan skew group. Scan skew group names are used by the **report_scan_skew_group**, **remove_scan_skew_group**, and **preview_dft** commands.

-include_elements *object_list*

Specifies the cells to include in the scan skew group. The object list can contain leaf cells, hierarchical cells, and CTL-modeled cells. Hierarchical cells include all scan cells in their hierarchy. CTL-modeled cells include all their scan segments. Wildcards and collections are supported.

DESCRIPTION

You can use scan skew groups to provide manual guidance for lock-up latch insertion. A scan skew group is a group of scan cells that might have a different clock latency characteristic than other parts of the design. The DFT architect treats the scan skew group as if it were a unique clock domain.

Scan skew groups override the normal scan clock domain identification behaviors such as

- Scan clock name
- The **-internal_clocks** option of the **set_scan_configuration** command

- The **-mix_internal_clock_driver** option of the **set_scan_configuration** command

If clock mixing is disabled, the cells in a scan skew group are not mixed in the same scan chain as other scan cells. If clock mixing is enabled, the cells in a scan skew group can be mixed in the same scan chain as other scan cells, with lock-up latches inserted as needed.

EXAMPLES

The following example defines a scan skew group for a particular design register:

```
prompt> set_scan_skew_group MY_GROUP \  
-include {CONFIG_reg[*]}
```

SEE ALSO

remove_scan_skew_group(2)
report_scan_skew_group(2)

set_scan_state

Sets the scan state status for the current db design.

SYNTAX

```
int set_scan_state  
    unknown | test_ready | scan_existing
```

ARGUMENTS

DESCRIPTION

This command sets the scan state status for the current design. Valid values are **unknown** (the scan state of the design is unknown), **test_ready** (the design is scan-replaced), and **scan_existing** (the design is scan-inserted).

Use this command only on a design that has been scan-replaced using, for example, **compile -scan**, so that the Q outputs of scan flip-flops are connected to the scan inputs and the scan enable pins are connected to logic zero. If there are nonscan elements in the design, use **set_scan_element false** to identify them.

This command can modify the netlist in memory if synchronous-logic shift register reidentification is enabled. For details, see the man page for the **compile_seqmap_identify_shift_registers_with_synchronous_logic_ascii** variable.

EXAMPLES

The following example illustrates setting the test_ready scan state status for the current design.

```
prompt> set_scan_state test_ready
```

SEE ALSO

```
current_design(2)  
insert_dft(2)  
reset_design(2)  
set_scan_configuration(2)
```

```
set_scan_element(2)  
set_scan_path(2)
```

set_scan_suppress_toggling

Specifies how the **insert_dft** command should insert gating at scan flip-flop functional outputs to suppress downstream toggling activity during scan shift.

SYNTAX

```
status set_scan_suppress_toggling  
[-selection_method manual | auto | mixed]  
[-include_elements cell_design_ref_list]  
[-exclude_elements cell_design_ref_list]  
[-min_slack minimum_timing_slack_after_gating]  
[-ignore_timing_impact true | false]  
[-total_percentage_gating percentage_value]
```

Data Types

<i>cell_design_ref_list</i>	list
<i>minimum_timing_slack_after_gating</i>	float
<i>percentage_value</i>	float

ARGUMENTS

-selection_method manual | auto | mixed

Specifies the scan flip-flop selection method used by the **insert_dft** command. The default is **manual** which applies gating only for objects manually specified with the **-include_elements** option. The **auto** value enables automatic selection of flip-flops for gating using power-based heuristics. The "mixed" value enables a combined approach, allowing the manual specification to be supplemented with automatic selection by the tool.

-include_elements *cell_design_ref_list*

Specifies leaf cells, hierarchical cells, or designs that should be gated. Hierarchical cells and designs implicitly include all leaf scan cells within the block or design.

Wildcards and collections are supported.

-exclude_elements *cell_design_ref_list*

Specifies leaf cells, hierarchical cells, or designs that should be excluded from gating. Hierarchical cells and designs implicitly include all leaf scan cells within the block or design.

Wildcards and collections are supported.

-min_slack *minimum_timing_slack_after_gating*

Specifies the required minimum slack value after gating is added. Valid values are between 0 and 1000. The default is 0, which

means the slack should not be negative after gating is added. This option is ignored if **-ignore_timing_impact true** has been specified.

-ignore_timing_impact true | false

Indicates whether slack should be ignored when determining which scan cells will be gated. The default is false.

-total_percentage_gating percentage_value

Specifies the target percentage of scan flip-flops to be automatically selected for gating when the **-selection_method** option is set to **auto** or **mixed**. The default for this option is 5.

DESCRIPTION

During scan testing, when scan data shifts through the scan chains, the scan flip-flop functional output pins are toggling. This causes switching activity in downstream functional logic, which can increase the power dissipation during scan shift.

To reduce power consumption, the **insert_dft** command can insert gating logic which suppresses the switching activity of selected scan flop functional output pins when the scan enable signal is asserted. The method used to select functional output gating locations is configured with the **set_scan_suppress_toggling** command.

The **-selection_method** option specifies the method used to select scan cells for gating. The default is **manual** which applies gating only for objects manually specified with the **-include_elements** option. If no **set_scan_suppress_toggling** command has been specified, the default behavior is that no scan cells are gated because no flip-flops have been manually specified.

When the **-selection_method** option is set to **auto**, scan cells are automatically selected for gating using propagated power analysis to maximize the power savings. The desired percentage of gated scan flops is specified with the **-total_percentage_gating** option. A percentage value between 5% and 30% is typically specified. Tradeoffs between test mode power consumption, functional power consumption, and functional timing must be considered when choosing a gating percentage value. As the gating percentage value is increased, leakage power increases, and the potential for layout congestion and timing closure difficulty increases. While considering timing and power tradeoffs, you should also consider that TetraMAX ATPG has several power-aware algorithms that seek to reduce shift and capture flop toggle rates during the ATPG process.

Setting the **-selection_method** option to **mixed** enables a combined approach, allowing the manual specification to be supplemented with automatic selection by the tool. All flip-flops specified with the **-include_elements** option are gated, even if the **-total_percentage_gating** target is exceeded. If the **-total_percentage_gating** target is not met by the manually specified flip-flops, additional flip-flops are selected automatically to meet the goal.

When the **-selection_method** option is set to **manual** or **mixed**, the set of manually-selected scan cells for output gating is specified with the **-include_elements** option. The **-include_elements** option is not supported in the **auto** selection mode.

Specific scan cells can be excluded from output gating with the **-exclude_elements** option. This option is supported for all selection methods.

By default, output gating is only inserted at a functional output if the resulting slack would be nonnegative. If a higher positive slack threshold is desired for selection, it can be specified with the **-min_slack** option. Alternatively, the slack requirement can be disregarded altogether with the **-ignore_timing_impact** option, which allows gating to be inserted even at negative slack functional outputs.

A flip-flop is excluded from output gating if it meets any of the following criteria:

- The flip-flop is manually excluded with the **-exclude_elements** option.
- The flip-flop is part of a shift register segment identified by the **compile_ultra** command.
- In a bottom-up flow, the flip-flop is within a block where test models are used or you have specified a **set_dont_touch** attribute

on the block.

- The Q or QN pin of the flip-flop connects only to the scan-in signal. (They can be gated if they are functionally connected.)
- The logic does not meet the minimum slack limit or would not meet the minimum slack limit if gating is inserted.

When the selection mode is set to auto or mixed, the **insert_dft** command reports the number of scan flip-flops selected for gating, the number considered for gating, the number that you manually included, and the number rejected due to timing considerations.

By default, the toggle suppression logic uses the driving scan cell's scan-enable signal. To define a dedicated scan-enable signal for suppression, define the signal using the **-usage scan_toggle** option of the **set_dft_signal** command, as shown in the examples.

Use the **report_scan_suppress_toggling** command to confirm the option settings you specified with the **set_scan_suppress_toggling** command. Use the **remove_scan_suppress_toggling** command to remove the previous toggling suppression settings applied.

EXAMPLES

The following example specifies that the **insert_dft** command should insert power gating on the optimal 10% of scan cell functional output pins:

```
prompt> set_scan_suppress_toggling -selection_method auto -total_percentage_gating 10
```

The following example specifies that the **insert_dft** command should insert power gating on the optimal 20% of scan cell functional output pins, without violating a minimum slack requirement:

```
prompt> set_scan_suppress_toggling -selection_method auto -total_percentage_gating 20 -min_slack 0.5
```

The following example manually specifies that all scan flip-flops in hierarchical cell BLK should be gated by the **insert_dft** command, except for some critical output scan flip-flops which should not be gated:

```
prompt> set_scan_suppress_toggling -include_elements {BLK} -exclude_elements {BLK/OUT_reg*}
```

The following example specifies that the **insert_dft** command should insert power gating on the functional output pins of all valid scan flip-flops within the MULTACCUM design:

```
prompt> set_scan_suppress_toggling -include_elements [get_designs MULTACCUM]
```

The following example shows a usage of mixed selection method:

```
prompt> set_scan_suppress_toggling -selection_method mixed -total_percentage_gating 20 -include_elements [get_designs MULTACCUM]
```

The following example shows how to create and configure a new SE signal **tog_se**, to be used as exclusive toggling suppression controller for the cell **U1/rst_q1_reg**:

```
prompt> create_port -dir in tog_se
Creating port 'tog_se' in design 'CORE'
1
```

```
prompt> set_dft_signal -type ScanEnable -port tog_se -usage scan_toggle
Accepted dft signal specification for modes: all_dft
```

```
prompt> set_scan_suppress_toggling -include_elements [get_cells U1/rst_q1_reg]
Accepted set_scan_suppress_toggling specification.
1
```

SEE ALSO

insert_dft(2)
set_dft_signal(2)
remove_scan_suppress_toggling(2)
report_scan_suppress_toggling(2)

set_scenario_options

Sets the scenario options for one or more scenarios.

SYNTAX

```
status set_scenario_options
  [-scenarios scenario_list]
  [-setup true | false]
  [-hold true | false]
  [-leakage_power true | false]
  [-dynamic_power true | false]
  [-cts_mode true | false]
  [-cts_corner min | max | min_max | none]
  [-reset_all true | false]
```

Data Types

scenario_list list

ARGUMENTS

-scenarios *scenario_list*

Specifies the list of scenarios to which to apply the scenario options.

If you do not specify this option, the command applies the scenario options to the current scenario.

-setup true | false

Controls whether the setup (maximum delay) cost is enabled for the specified scenarios.

The default is **true**. When set to **false**, optimization ignores setup violations in the specified scenarios.

-hold true | false

This option is not used by Design Compiler. If the design is saved with this setting, it is saved in the .ddc file and can be used by tools that read the .ddc file later in the flow.

-leakage_power true | false

Controls whether the leakage power cost is enabled for the specified scenarios.

The default is **false**.

If set to **true**, it is incompatible with the %LVT cells constraint (which is scenario independent).

-dynamic_power true | false

If set to true, low power placement must also be enabled (by setting the flag **power_low_power_placement**) to true to be effective. Note that low power placement requires **compile_ultra -spg**.

-cts_mode true | false

This option is not used by Design Compiler. If the design is saved with this setting, it is saved in the .ddc file and can be used by tools that read the .ddc file later in the flow.

-cts_corner min | max | min_max | none

This option is not used by Design Compiler. If the design is saved with this setting, it is saved in the .ddc file and can be used by tools that read the .ddc file later in the flow.

-reset_all true | false

When set to **true**, resets all scenario options to default values. When **false** (the default), the scenario options remain unchanged.

DESCRIPTION

This command sets options to restrict a scenario to be used for specific costs. Combinations of the **-setup**, **-hold**, and **-leakage_power** options can be used to restrict a scenario. All scenario options can be applied to active and inactive scenarios.

Use the **report_scenario_options** command to report the options set by this command.

Multicorner-Multimode Support

By default, this command applies to the current scenario. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following example uses the **set_scenario_options** command to restrict the MODE1 scenario for use only for setup timing and uses the **-dynamic_power** option to enable dynamic power optimization.

```
prompt> create_scenario MODE1
Warning: Discarding all scenario specific information previously
defined in this session.
(UUID-1008)
Current scenario is: MODE1
1

prompt> set_scenario_options -setup true \
-dynamic_power true -leakage_power false
1

prompt> report_scenario_options

*****
Report : scenario options
Design : TEST03
Version: C-2009.06
Date   : Wed Mar 19 14:57:08 2008
```

Scenario: MODE1 is active.

setup : false

hold : true

leakage_power : false

dynamic_power : true

SEE ALSO

report_scenario_options(2)

create_scenario(2)

current_scenario(2)

report_scenarios(2)

set_multi_vth_constraint(2)

set_fix_hold(2)

set_scope

Specifies the current scope.

SYNTAX

```
string set_scope  
[instance]
```

Data Types

instance string

ARGUMENTS

instance

Specifies the working instance in dc_shell.

- If *instance* is not specified, the focus returns to the top level of the current design.
- If *instance* is ".", the tool returns to the working instance.
- If *instance* is "..", the context is moved up one level in the hierarchy of the current design.
- If *instance* begins with "/", the tool returns to the working instance of the design whose name is after the "/".
- Multiple levels of hierarchy can be traversed in a single call to the **set_scope** command, by separating the cell names with a slash (/).

More complex examples of *instance* arguments are described in the EXAMPLES section below.

DESCRIPTION

The **set_scope** command sets the current scope. Functionally, this command is a subset of the **current_instance** command, but has a different return value. Upon success, the **set_scope** command returns the current scope prior to the execution of this command as a full path string relative to the current design. The command returns a null string upon failure.

The **set_scope** command does not work on leaf cells. If you attempt to run the **set_scope** command on a leaf cell, the tool issues the following error message:

Error: A leaf cell instance A/B/leaf has been specified as the scope. (UPF-012)

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the scope to the U1/U2 hierarchy, if the U1 cell exists in the current hierarchy and the U2 cell exists in the U1 cell.

```
prompt> set_scope U1/U2
```

The following example sets the scope to two levels of hierarchy above the current instance, and then down one level to the MY_INST cell.

```
prompt> set_scope ../../MY_INST
```

SEE ALSO

`current_design(2)`
`current_instance(2)`

set_script_runtime_report_mode

Sets or queries the report mode of script runtime.

SYNTAX

```
string set_script_runtime_report_mode  
[mode_name]
```

ARGUMENTS

mode_name

Specifies the name of the new mode. The supported mode names are **off**, **summary**, and **verbose**. If you do not specify a mode name, the command returns the current mode name.

DESCRIPTION

The **set_script_runtime_report_mode** command sets the new mode for script runtime report or queries the current report mode. When the script runtime reporting is turned on, possible runtime saving in the script, such as avoiding unnecessary changes because of the **current_design** command, are reported.

You can use this command in de_shell only.

EXAMPLES

The following examples show how to use the command:

```
prompt> set_script_runtime_report_mode  
off
```

```
prompt> set_script_runtime_report_mode verbose  
verbose
```

SEE ALSO

SCR-01(n)

SCR-02(n)

SCR-03(n)

set_self_gating_objects

Forces the enabling or disabling of self gating for specified objects in the current design, overriding all conditions necessary for automatic self gating, by the **compile_ultra -self_gating** command. Objects can be of type register, hierarchical cell, power domain or design.

SYNTAX

```
status set_self_gating_objects  
  [-force_include object_list]  
  [-exclude object_list]  
  [-include object_list]  
  [-undo object_list]
```

Data Types

object_list list

ARGUMENTS

-force_include *object_list*

Specifies a list of objects in the current design to be self gated, overriding the conditions set by the **set_self_gating_options** command.

-exclude *object_list*

Specifies a list of objects in the current design to be excluded from self gating.

-include *object_list*

Specifies a list of objects in the current design for which self gating should be done as per the options specified by the **set_self_gating_options** command. This is the default for all objects in the design.

-undo *object_list*

Specifies a list of objects in the current design for which the inclusion or exclusion criteria for self gating should be removed

The *object_lists* are mutually exclusive, that is, an object cannot be specified with more than one of the following options: -**force_include**, -**exclude**, -**include** or -**undo**.

DESCRIPTION

This command specifies the objects on which self gating is to be either enabled or disabled on its registers, overriding all conditions necessary for automatic self gating by the **compile_ultra -self_gating** command.

For objects specified with the **-force_include** option, self gating is performed regardless of the previous specification. Objects specified with the **-exclude** option are excluded from self gating regardless of the previous specification.

Excluding a self-gated register does not cause the register to be ungated in a incremental call of **compile_ultra -self_gating**

Use the **-include** option to specify that regular self-gating criteria should be used on the specified objects. This option is used to explicitly set an object as included for self gating when one of its higher hierarchies is forcefully included or excluded. Achieving a donut shaped region, where regular self-gating is performed, in the middle of hierarchies completely excluded or forcefully included.

Use the **-undo** option to remove a previously specified criteria on the specified objects.

You must run the **set_self_gating_objects** command before running the **compile_ultra -self_gating** command. The specifications persists during the subsequent executions of the **compile_ultra -self_gating**.

EXAMPLES

The following example excludes the register bank A_reg in the current design, from self gating:

```
prompt> set_self_gating_objects -exclude A_reg[*]
```

In the following example, the register bank D_OUT_reg in the MID sub design, is self gated ignoring the previous specification.

```
prompt> set_self_gating_objects -force_include MID/D_OUT_reg[*]
```

The following example includes and excludes certain registers from self gating in the ADDER sub design:

```
prompt> set_self_gating_objects \  
-force_include ADDER/out1_reg[*] \  
-exclude ADDER/out2_reg[*]
```

The following example removes the directive to include and exclude self gating for the registers in the ADDER sub design:

```
prompt> set_self_gating_objects \  
-undo {ADDER/out1_reg[*] ADDER/out2_reg[*]}
```

The following example excludes every register present in the ADDER sub design, except for the out1_reg bank, which is self gated according to the specified options.

```
prompt> set_self_gating_objects \  
-exclude ADDER \  
-include ADDER/out1_reg[*]
```

SEE ALSO

compile_ultra(2)
set_self_gating_options(2)
report_self_gating(2)
set_clock_gating_objects(2)

set_self_gating_options

Sets the self-gating options for the self-gate insertion. This command is supported only in topographical mode.

SYNTAX

```
status set_self_gating_options
  [-min_fanout min_fanout_count]
  [-max_fanout max_fanout_count]
  [-min_bitwidth min_bitwidth_count]
  [-max_bitwidth max_bitwidth_count]
  [-interaction_with_clock_gating none | insert | merge]
```

Data Types

<i>min_fanout_count</i>	integer
<i>max_fanout_count</i>	integer
<i>min_bitwidth_count</i>	integer
<i>max_bitwidth_count</i>	integer

ARGUMENTS

-min_fanout *min_fanout_count*

This option is deprecated; instead, use "-min_bitwidth". Specifies the minimum bitwidth of a single self-gating cell.

-max_fanout *max_fanout_count*

This option is deprecated; instead, use "-max_bitwidth". Specifies the maximum bitwidth of a single self-gating cell.

-min_bitwidth *min_bitwidth_count*

Specifies the minimum bitwidth of a single self-gating cell.

-max_bitwidth *max_bitwidth_count*

Specifies the maximum bitwidth of a single self-gating cell.

-interaction_with_clock_gating none | insert | merge

Specifies the interaction between the self-gates to be inserted and traditional clock-gates.

The value **none** prevents the tool from inserting self-gates on already clock-gated registers.

The value **insert** directs the tool to insert self-gates on already clock-gated registers.

The value **merge** directs the tool to insert self-gates on already clock-gated registers and merges the clock gates and the self-gates. Merging can happen only if the clock gate and the self-gate are in the same hierarchy, have same operating condition, and same test enable signal. Merging can be performed in a compile incremental as a post process of the self-gating insertion.

DESCRIPTION

This command sets the self-gating options to be used for self-gating with the **compile_ultra -self_gating** command.

The self-gating options specifies the following aspects of self-gating:

- Conditions when self-gating is applied
- Interaction with traditional clock-gating elements

EXAMPLES

Default values are assumed for all unspecified options. In particular, option values from previous self-gating options settings are not stored.

The following example sets the self-gating options such that each self-gate is inserted to gate at least 2 and at most 10 bit-widths that belong to single-bit and/or multibit registers:

```
prompt> set_self_gating_options -min_bitwidth 2 -max_bitwidth 10
```

The following example sets the self-gating options such that traditional clock gated registers are allowed to be self-gated and, if possible, both gates are merged

```
prompt> set_self_gating_options -interaction_with_clock_gating merge
```

SEE ALSO

compile_ultra(2)
report_self_gating(2)
all_self_gates(2)

set_sense

Specifies unateness propagating forward for pins with respect to clock source.

SYNTAX

```
status set_sense
  [-type clock]
  [-positive]
  [-negative]
  [-stop_propagation]
  [-pulse pulse_type]
  [-clocks clock_list]
  pins
  -clock_leaf
```

Data Types

```
pulse_type  string
clock_list  list
pins        list
```

ARGUMENTS

-type clock

Currently the sense is only applied to clock networks. If you do not specify this option, the default is **clock**.

-positive

Propagates only the positive unate paths forward from the specified pins.

The **-positive** option is mutually exclusive with the **-negative**, **-pulse**, **-logical_stop_propagation**, and **-stop_propagation** options. You must specify one of these options.

-negative

Propagates only the negative unate paths forward from the specified pins.

The **-negative** option is mutually exclusive with the **-positive**, **-pulse**, **-logical_stop_propagation**, and **-stop_propagation** options. You must specify one of these options.

-stop_propagation

Stops propagation of the specified clocks at the specified pins along the clock paths only.

The **-stop_propagation** option is mutually exclusive with the **-positive**, **-negative**, **-logical_stop_propagation**, and **-pulse**

options. You must specify one of these options.

-pulse *pulse_type*

Specifies the type of pulse clock that is generated from the specified pins. You must specify one of the following values: **rise_triggered_high_pulse**, **rise_triggered_low_pulse**, **fall_triggered_high_pulse**, or **fall_triggered_low_pulse**.

The **-pulse** option is mutually exclusive with the **-positive**, **-negative**, **-logical_stop_propagation**, and **-stop_propagation** options. You must specify one of these options.

-clocks *clock_list*

Specifies the clocks to which the specified clock sense applies.

If you do not specify this option, the clock sense setting applies to any clock that passes through the specified pins.

pins

Specifies the pins on which to set the clock sense.

DESCRIPTION

This command provides the capability to define the clock sense at nonunate points in the clock network. The command is ignored if it is used on a unate point in the clock network. The specified clock sense propagates forward from the specified pins.

If the **-clocks** option is used, only the specified clocks are considered. Otherwise, all clocks passing through the specified pins are considered.

Warning messages are issued if the specified sense cannot be respected on the specified pins. Hierarchical pins are not supported.

To undo the settings made by the **set_sense** command, use the **remove_sense** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example specifies positive unateness for a pin named XOR/Z with respect to the CLK1 clock:

```
prompt> set_sense -positive -clocks [get_clocks CLK1] XOR/Z
```

The following example specifies a pulse type of **rise_triggered_high_pulse** for all clocks passing through a pin named MUX/Z:

```
prompt> set_sense -pulse rise_triggered_high_pulse MUX/Z
```

SEE ALSO

[remove_sense\(2\)](#)

set_separate_process_options

Sets options that control whether the tool uses separate UNIX processes for extraction, placement, and routing.

SYNTAX

status **set_separate_process_options**
[-placement true | false]

ARGUMENTS

-placement true | false

Specifies whether placement should use a separate UNIX process.

If true, a separate UNIX process is used.

If false (the default), placement is run within the parent UNIX process.

DESCRIPTION

The **set_separate_process_options** command sets options that control whether the tool uses separate UNIX processes for extraction, placement, and routing. The setting is not saved in database. It is applicable only in the current session and must be reapplied if desired in a new session.

You can use the **report_separate_process_options** command to report the current settings for separate processes.

EXAMPLES

The following example sets the **-placement** option to true to turn on separate processes for placement.

```
prompt> set_separate_process_option -placement true  
1
```

SEE ALSO

report_separate_process_options(2)

set_serialize_configuration

Specifies the serializer configuration for the design.

SYNTAX

```
status set_serialize_configuration
  [-inputs number_of_deserializer_inputs]
  [-outputs number_of_serializer_outputs]
  [-ip_inputs list_core_instance_and_number_of_deserializer_ip_inputs]
  [-ip_outputs list_core_instance_and_number_of_serializer_ip_outputs]
  [-update_stage true | false]
  [-exclude_clocks list_of_clock_names_to_exclude]
  [-serializer_clock clock_for_serializer_controller]
  [-update_clock clock_for_update_stage_in_load_deserializer]
  [-strobe user_defined_signal_for_strobe_in_unload_serializer]
  [-wide_duty_cycle true | false]
  [-parallel_mode parallel_scan_compression_mode_name]
  [-test_mode serial_scan_compression_mode_name]
```

ARGUMENTS

-inputs *number_of_deserializer_inputs*

Specifies the number of inputs used to load scan data in the deserializer for scan compression. The inputs are a subset of the scan inputs used for base mode (reconfigurable scan mode).

The default is equal to the minimum number of chains that DFT would build if scan compression were not enabled.

-outputs *number_of_serializer_outputs*

Specifies the number of outputs used to observe scan data from the serializer for scan compression. The outputs are a subset of the scan outputs used for base mode (reconfigurable scan mode).

The default is equal to the minimum number of chains that DFT would build if scan compression were not enabled.

-ip_inputs *list_core_instance_and_number_of_deserializer_ip_inputs*

In the serializer IP insertion flow, this command accepts a list of pairs of the core instance names and the number of deserializer inputs for the core. The inputs are a subset of the scan inputs used for base mode (reconfigurable scan mode).

The default is null (no value specified). Deserializer IP is inserted only for the DFTMAX cores specified in the list.

-ip_outputs *list_core_instance_and_number_of_serializer_ip_outputs*

In serializer IP insertions flow, this command accepts list of pairs of the core instance names and the number of serializer outputs for the core. The inputs are a subset of the scan outputs used for base mode (reconfigurable scan mode).

The default is null (no value specified). Serializer IP is inserted only for the DFTMAX cores specified in the list.

-update_stage true | false

Adds an update stage to the deserializer.

The default is **false**.

-exclude_clocks *list_of_clock_names_to_exclude*

Specifies the list of clock names to be excluded from gating in the serializer clock controller. The specified clock must be previously declared as a test clock using the **set_dft_signal** command.

The default is null (no value specified).

-serializer_clock *clock_for_serializer_controller*

In hierarchical flows, when **set_scan_compression_configuration -serialize** is set to **core_level**, you can specify the clock to be used for the deserializer-serializer. The specified clock must be previously declared as a test clock using the **set_dft_signal** command.

The default is null (no value specified).

-update_clock *clock_for_update_stage_in_load_deserializer*

In hierarchical flows, when the **set_scan_compression_configuration -serialize** option is set to **core_level**, you can specify the clock to be used for the update stage in the unload deserializer. The specified clock must be previously declared as a test clock using the **set_dft_signal** command.

The default is null (no value specified).

-strobe *user_defined_signal_for_strobe_in_unload_serializer*

In hierarchical flows, when **set_scan_compression_configuration -serialize** is set to **core_level**, you can specify a test data signal to be used for the strobe signal in load serializer. The specified signal must be previously declared as a test data signal using the **set_dft_signal** command.

The default is null (no value specified).

-wide_duty_cycle true | false

When set to **true**, the serializer clock controller generates the internally generated scan shift clocks and the update stage clock with a duty cycle closer to 50%, which helps prevent timing issues.

The default is **false**, which generates these clocks by passing through selected high-frequency clock pulses from the clock source.

-parallel_mode *parallel_scan_compression_mode_name*

Specifies the parallel ScanCompression mode that is to be associated with the serial ScanCompression_mode as indicated by the **-test_mode** switch.

The default is null (no value specified).

-test_mode *serial_scan_compression_mode_name*

Specifies the scan compression test mode to be applied to the appropriate parameters of this serializer configuration specification.

The default is *all_dft*, except when **define_test_mode** is used. In this case, the default is the value of the test mode defined last.

DESCRIPTION

The **set_serialize_configuration** command allows you to specify certain DFTMAX insertion parameters prior to using the **insert_dft** command.

The command also enables the configuration of the number of serial scan inputs and serial scan outputs that are to be used for a particular serial scan compression mode. In the serializer IP insertion flows, you can specify the number of serial scan inputs and outputs for every DFTMAX core using the **-ip_inputs** and **-ip_outputs** options.

The other parameters are used to specify the clocks that can be excluded from clock-gating using the **-exclude_clocks** option, as well as the serializer clock in core-level flows using the **-serializer_clock** option.

An update stage can be inserted along with the deserializer to enable fast clocking schemes when the **-update_stage** option is set to **true**. A parallel scan compression mode can be associated with the serial scan compression mode specified by the **-test_mode** option using the **-parallel_mode** option. The parallel mode has direct access to the decompressor-compressor active in the serial scan compression mode.

SEE ALSO

define_test_mode(2)
insert_dft(2)
preview_dft(2)
report_serialize_configuration(2)
reset_serialize_configuration(2)

set_size_only

Controls whether the specified leaf cells can only be sized during optimization.

SYNTAX

```
status set_size_only  
[-all_instances]  
object_list  
[true | false]
```

Data Types

object_list collection

ARGUMENTS

-all_instances

Controls how the size-only setting is applied to child instances of multiply instantiated designs.

When you use this option, if a specified cell is in a multiply instantiated instance of a subdesign, the tool applies the size-only setting to the similar cells in all instances of that subdesign.

This is especially useful to avoid changing the current design to issue this command in different designs.

object_list

Specifies the leaf cells on which to apply the size-only setting.

true | false

Specifies the value of the size-only setting.

The default is **true**, so **set_size_only U1** has the same effect as **set_size_only U1 true**.

DESCRIPTION

This command applies size-only settings to the specified leaf cells.

When you apply a size-only setting of **true** to leaf cell, optimization can perform only sizing operations on that cell. A setting of **false** removes the size-only settings for the specified leaf cells.

For example, if apply a size-only setting of **true** to a cell, the tool attempts to optimize this cell by replacing it with a cell of the same

function. The size-only setting on a cell does not prevent changes to the net driven by this cell. For example buffers can be added to this net. To prevent changes to the net, use the **set_dont_touch** command to set the **dont_touch** attribute on the net.

To report the size-only settings, use the **report_cell** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example applies a size-only setting of **true** to the cell named U1.

```
prompt> set_size_only U1
```

The following example removes the size-only setting from the cell named U1.

```
prompt> set_size_only U1 false
```

The following example uses the **-all_instances** option.

```
prompt> set_size_only -all_instances mid1/bot1/c1
Information: Set size_only for all instances of cell 'c1'
in subdesign 'bot'. (UID-193)
1
```

SEE ALSO

- compile(2)
- get_attribute(2)
- report_attribute(2)
- report_cell(2)
- set_compile_directives(2)

set_streaming_compression_configuration

Specifies the streaming compression configuration for the design.

SYNTAX

```
status set_streaming_compression_configuration
[-compressed_max_length max_chain_length
| -max_length max_chain_length
| -chain_count chain_count]
[-inputs number_of_inputs]
[-outputs number_of_outputs]
[-external_clock_chain true | false]
[-test_mode compression_mode_name]
[-base_mode base_mode_name]
[-clock clock_name]
[-decompressor_clock clock_name]
[-compressor_clock clock_name]
[-exclude_clocks clock_name_list]
[-min_power false | true]
[-shift_power_groups false | true]
[-shift_power_chain_length chain_length]
[-shift_power_chain_ratio chain_ratio]
[-shift_power_clock clock_name]
[-shift_power_disable test_control_name]
[-configuration_ports Specify_ports_for_multiple_configuration_IP]
```

Data Types

<i>max_chain_length</i>	integer
<i>chain_count</i>	integer
<i>number_of_inputs</i>	integer
<i>number_of_outputs</i>	integer
<i>compression_mode_name</i>	string
<i>base_mode_name</i>	string
<i>clock_name</i>	string
<i>clock_name_list</i>	list
<i>chain_length</i>	integer
<i>chain_ratio</i>	integer
<i>test_control_name</i>	string

ARGUMENTS

-compressed_max_length *max_chain_length*

Specifies the maximum allowed number of shift cycles of the entire scan compression path (from decompressor inputs to compressor outputs). The tool adjusts the codec shift register and compressed chain lengths together to find an optimal architecture that meets this constraint. If unable to do so, the tool issues an error. See the DESCRIPTION section for more information.

This specification applies only to the codec and its compressed chains. External scan constructs, such as scan data pipelining or external chains, are not included.

-max_length max_chain_length

Specifies the maximum allowed length of the compressed scan chains only (not including the codec shift registers). The tool attempts to meet this constraint. If unable to do so, the tool issues a warning. See the DESCRIPTION section for more information.

-chain_count chain_count

Specifies the number of compressed scan chains to build. The tool attempts to meet this constraint. If unable to do so, the tool issues a warning. See the DESCRIPTION section for more information.

-inputs number_of_inputs

Specifies the number of scan data inputs used to drive the streaming compression decompressor.

-outputs number_of_outputs

Specifies the number of scan data outputs used to capture data from the streaming compression compressor.

-external_clock_chain true | false

Controls whether an external on-chip clocking (OCC) chain is created.

When set to **true** (the default), an external clock chain is created.

When set to **false**, the clock chain is driven by the decompressor. In this case, you must have at least two inputs to the streaming compression codec (an **-inputs** value of 2 or more).

-test_mode compression_mode_name

Specifies the streaming compression test mode to be applied to the appropriate parameters of this streaming configuration specification.

The default value is **all_dft**, except when **define_test_mode** is used. In this case, the default is the value of the **test_mode** defined last.

-base_mode base_mode_name

Specifies the base mode (or Internal_scan mode) that is to be associated with the ScanCompression_mode indicated by the **-test_mode** switch. By default, the value is **Internal_scan**.

-clock clock_name

Specifies the scan clock to be used for the streaming compression codec (compressor and decompressor) to be inserted. The specified clock must be previously declared as a test clock using the **set_dft_signal** command. By default, the tool selects the dominant scan clock, which typically results in the fewest lockup latches at the decompressor outputs and compressor inputs.

In an OCC controller flow (DFT-inserted or user-defined), you can specify the name of an ATE clock that is predefined with the **set_dft_signal** command. This specification is mapped to an OCC controller clock associated with that ATE clock during DFT insertion.

-decompressor_clock clock_name

Specifies the scan clock to be used for the streaming decompressor to be inserted. This option is similar to the **-clock** option, but it specifies the clock for the decompressor only. This option takes precedence over the **-clock** option.

-compressor_clock *clock_name*

Specifies the scan clock to be used for the streaming compressor to be inserted. This option is similar to the **-clock** option, but it specifies the clock for the compressor only. This option takes precedence over the **-clock** option.

-exclude_clocks *clock_name_list*

Specifies the list of clock names to be excluded from automatic codec clock selection (when the **-clock** option is not used). The specified clocks must be previously declared as test clocks using the **set_dft_signal** command. The default is an empty list.

-min_power *false | true*

When set to **true**, specifies that compressor inputs are to be gated for power saving. The **insert_dft** command creates the compressor logic such that it is active only during scan shift in that compression mode. The compressor logic is inactive during scan capture in that mode, during other test modes, and during mission mode. This eliminates power consumption due to switching activity in the compressor during functional operation (which is at-speed rather than at test clock frequencies). The default of this option is **false**.

-shift_power_groups *false | true*

Enables the shift power groups feature. This feature implements a shift power control chain that allows ATPG to selectively gate groups of compressed chains for shift power reduction. The gated chains still shift, but they shift constant (non-toggling) values into the chains, which reduces toggle activity.

If the option is set true; DFTMAX Ultra calculates shift power chain length automatically. DFTMAX Ultra calculates SPC length as: Minimum of {number of chains of codec/3, max shift length of chain}.

For a specific SPC length or chain ratio use **-shift_power_chain_length** or **-shift_power_chain_ratio** options.

The default is **false**, which does not implement the control chain or gating logic.

-shift_power_chain_length *chain_length*

Specifies the length of the shift power control (SPC) chain. This value directly controls the number of compressed chain groups.

Use this option to keep the length of the SPC chain constant as the compression architecture changes.

-shift_power_chain_ratio *chain_ratio*

Specifies the ratio of compressed chains to each shift power control (SPC) register. This value directly controls the number of compressed chains in each group.

Use this option to keep the group size constant as the compression architecture changes.

-shift_power_clock *clock_name*

Specifies the clock to use for the shift power control chain. The specified clock must be previously declared as a test clock using the **set_dft_signal** command. By default decompressor clock is selected as shift power clock.

-shift_power_disable *test_control_name*

Specifies the signal that, when asserted to its active value, disables the shift power logic. The specified signal must be previously declared as a TestControl signal using the **set_dft_signal** command. By default, the shift power logic is always active and no shift power disabling logic is implemented.

DESCRIPTION

The **set_streaming_compression_configuration** command allows you to specify certain streaming compression insertion

parameters prior to using the **insert_dft** command.

There are three related options to control the number of compressed scan chains. Only one option can take effect at a time. They are, in order of highest precedence first:

- **-compressed_max_length**
- **-max_length**
- **-chain_count**

The command also enables configuration of the number of scan inputs (using **-inputs**) and scan outputs (using **-outputs**).

By default, the tool selects clocks for the decompressor and compressor that minimize the number of lock-up latches. (For most designs, the decompressor and compressor use the same clock.) You can use the following options to control codec clock selection, in order of highest precedence first:

- **-decompressor_clock, -compressor_clock**
- **-clock**
- **-exclude_clocks** (applies to automatic clock selection)
- Automatic clock selection

There are five related options to configure shift power groups. They are:

- **-shift_power_groups**
- **-shift_power_chain_length**
- **-shift_power_chain_ratio**
- **-shift_power_clock**
- **-shift_power_disable**

SEE ALSO

define_test_mode(2)
insert_dft(2)
preview_dft(2)
report_streaming_compression_configuration(2)
reset_streaming_compression_configuration(2)

set_structure

Sets structure attributes on a design or on a list of designs, to determine how the designs are structured during compile.

SYNTAX

```
status set_structure  
[true | false]  
[-design design_list]  
[-boolean true | false]  
[-boolean_effort low | medium | high]  
[-timing true | false]
```

Data Types

design_list list

ARGUMENTS

true | false

Specifies the value to which the **structure** attribute is to be set. If set to **true** (the default), **compile** adds logic structure to the specified designs by adding intermediate variables factored from the designs' equations.

-design *design_list*

Specifies a list of designs on which the **structure** attribute is to be set. The default value is the current design.

-boolean true | false

Specifies the value to which the **structure_boolean** attribute is to be set. If set to **true**, and if the **structure** attribute is also set to **true**, **compile** uses Boolean (non-algebraic) optimization on the specified designs. The default value is **false**. If **structure** is **false**, **structure_boolean** is ignored.

-boolean_effort low | medium | high

Specifies the value to which the **boolean_effort** attribute is to be set. This attribute specifies the relative amount of CPU time to be spent by Boolean (non-algebraic) optimization during the structuring phase of **compile**. The default value is **low**.

-timing true | false

Specifies the value to which the **structure_timing** attribute is to be set. If **structure** is set to **true**, by default **structure_timing** is also **true**, and **compile** uses timing-driven structuring on the specified designs. Set this attribute to **false** if you do not want timing-driven structuring. If **structure** is **false**, **structure_timing** is ignored.

DESCRIPTION

The **set_structure** command sets structure attributes on a design or on a list of designs, to determine how the designs are structured during **compile**. If **set_structure** is called without a value, by default the design is structured.

This optimization step adds logic structure to a design by adding intermediate variables. The **compile** command searches for subfunctions that can be factored out and evaluates these factors based on the size of the factor and the number of times the factor appears in the design. The subfunctions that most reduce the logic are converted to intermediate variables and factored out of the design's equations. When **structure** is **true**, by default **structure_timing** is also **true**, and timing-driving structuring is automatically done unless **structure_timing** is set to **false**. With **structure** set to **true**, setting **structure_boolean** to **true** enables Boolean (non-algebraic) optimization during structuring. If **structure** is **false**, both **structure_boolean** and **structure_timing** are ignored.

To remove the **structure**, **structure_boolean**, **boolean_effort**, and **structure_timing** attributes, use the **remove_attribute** command. To remove all attributes, use the **reset_design** command.

EXAMPLES

In the following example, structuring, timing-driven structuring, and Boolean optimization are enabled on the design named TEST1 and structuring is disabled on the design named TEST2:

```
prompt> set_structure -design TEST1 -boolean
```

```
prompt> set_structure -design TEST2 false
```

In the following example, structuring is enabled and timing-driven structuring is disabled on the design named ADD1:

```
prompt> set_structure -design ADD1 -timing false
```

In the following example, structuring is disabled for all designs read in memory:

```
prompt> set_structure -design [get_designs *] false
```

SEE ALSO

- `compile(2)`
- `remove_attribute(2)`
- `reset_design(2)`
- `set_flatten(2)`
- `attributes(3)`

set_svf

Generates a Formality setup information file for efficient compare point matching in Formality.

SYNTAX

Boolean **set_svf**
 filename
 [-append]
 [-off]

ARGUMENTS

filename

Names the file into which Formality setup information will be recorded. You must specify a file name unless the -off option is specified.

-append

Appends to the specified file. If another Formality setup verification file is already open then it will be closed before opening the specified file. If -append is not used then **set_svf** will overwrite the named file, if it exists.

-off

Stops recording Formality setup information to the currently-open file. To resume recording into the same file, you must re-issue the **set_svf** command with the -append option.

DESCRIPTION

This command causes Design Compiler to start recording setup information for Formality, the Synopsys formal verification tool. The SVF ("Setup Verification for Formality") file that is produced is used by Formality during the matching step to facilitate the alignment of compare points. By using the automatically-generated SVF file, the user is relieved of the time-consuming and error-prone process of entering the setup information manually.

To record setup information for formal verification products other than Formality, use the *set_vsdc* command, instead.

The SVF file is used to account for name changes that occur during synthesis as a result of running commands such as group, ungroup, uniquify and change_names. Also, certain operations performed by the compile command perform transformations that are recorded in the SVF file.

Further information on SVF files can be found in the Formality User Guide.

Once this command is issued, Design Compiler will begin recording all relevant operations. Because information is buffered internally, the file may not be complete until recording is stopped. Recording is stopped by running "set_svf -off" or by exiting dc_shell. Running **set_svf** with a new file name will write out and close the original SVF file before starting the new one.

The SVF file is stored in encrypted format.

This command returns a Boolean value indicating whether **set_svf** succeeded (true) or not (false).

Formality setup recording might be enabled by default in Design Compiler's system-wide setup file. If so, then the file name used is "default.svf". If you prefer a different file name then you may run **set_svf** in your setup file or script prior to performing any recordable operations. Alternately, you may disable SVF recording completely by running "set_svf -off".

EXAMPLES

The following example shows a typical invocation of the command

```
prompt> set_svf my_design.svf
```

In the following example the Formality setup information is appended to the existing SVF file, barfile.svf.

```
prompt> set_svf -append barfile.svf
```

SEE ALSO

- change_names(2)
- compile(2)
- compile_ultra(2)
- group(2)
- set_vsdc(2)
- ungroup(2)
- uniquify(2)
- compile_seqmap_propagate_constants(3)
- fsm_auto_inferring(3)

set_switching_activity

Sets the switching activity annotation on nets, pins, ports and cells of the current design.

SYNTAX

```
status set_switching_activity
  [-static_probability static_probability]
  [-toggle_rate toggle_rate]
  [-state_condition state_condition]
  [-path_sources path_sources]
  [-rise_ratio rise_ratio]
  [-period period_value | -base_clock clock]
  [-type object_type_list]
  [-hierarchy]
  [object_list]
  [-verbose]
  [-scenarios {scenario_name1 scenario_name2 ... }]
```

Data Types

<i>static_probability</i>	float
<i>toggle_rate</i>	float
<i>state_condition</i>	string
<i>path_sources</i>	list
<i>rise_ratio</i>	float
<i>period_value</i>	float
<i>clock</i>	string
<i>object_type_list</i>	list
<i>object_list</i>	collection

ARGUMENTS

-static_probability *static_probability*

Specifies the static probability value. The static probability is a floating point number between 0.0 and 1.0. It represents the percentage of the time the signal is at logic 1. For example, a static probability value of 0.25 indicates that the signal is at logic 1 for 25% of the time.

-toggle_rate *toggle_rate*

Specifies the toggle rate value. The toggle rate is a positive floating point number that represents the number of 0->1 and 1->0 transitions that the signal makes during a period of time. The period is by default 1 unit and can be specified with the **-period** option. The time unit defined in the library is used as the time unit. Alternatively, a related clock can be annotated using the **-base_clock** argument, and the specified toggle rate will be relative to the related clock period.

-state_condition *state_condition*

Specifies the state condition when annotating state-dependent toggle rates on pins or state-dependent static probabilities on cells. State dependent toggle rates can be annotated when the internal power of the library cell pin is characterized with state-dependent power tables. State-dependent static probabilities can be annotated when the cell leakage power is characterized with state-dependent power tables. The state condition specified with this argument must be logically equivalent to a state condition in the internal or leakage power characterization.

-path_sources *path_sources*

Specifies the path sources when annotating path-dependent toggle rates on pins. This can be used when the library cell pin has path-dependent internal power characterization. The path sources specified with this argument must be the same as those in the internal power characterization.

-rise_ratio *rise_ratio*

Specifies the ratio of rise transitions to total transitions for the specified toggle rate when annotating pins that are characterized with both rise and fall internal power. The *rise_ratio* argument is a floating point number between 0.0 (all transitions are falling) and 1.0 (all transitions are rising). You need to specify a toggle rate in order to use this option. The default value is 0.5.

-period *period_value*

Specifies the time period for which the number of transitions given in the toggle rate *-toggle_rate* occur. The time units for this value are those specified in the main technology library. When this argument is used, the *-toggle_rate* specified by the **-toggle_rate** argument is divided by the given *-period_value*; The resulting toggle rate is then annotated to the object(s) specified with the **set_switching_activity** command. If the **-period** argument is not specified, a default value of 1.0 is assumed.

-base_clock *clock*

Specifies a clock to which *toggle_rate* is related. This clock is referred to as the object's related clock. When specified, the related clock is annotated to the design objects, and during power calculations the annotated toggle rate value is divided by the related clock period. The resulting toggle rate is then used for calculating power. The clock period can be changed between different runs of **report_power** and the resulting change in the toggle rate of the design objects will be used automatically without re-annotating the switching activity. You can also specify "" as the *clock* value of the **-base_clock** argument. This specifies that Power Compiler will infer the related clock on the design object automatically before power calculations.

-type *object_type_list*

Specifies a list of object types that will be used for implicitly selecting the objects that will be annotated with the specified switching activity information. The *object_type_list* argument can be a list of one or more of the following object types:

registers (sequential cell outputs),
tristates (tristate cell outputs),
black_boxes (black box cell outputs),
inputs (input design ports or hierarchical instance pins),
outputs (output design ports or hierarchical instance pins),
inouts (inout design ports / hierarchical instance pins),
ports (design port or hierarchical instance pins),
nets (nets),
regs_on_clocks *clock_list* (outputs of flip-flops clocked by the clocks in *clock_list*),
clock_gating_cells (clock gating cell outputs),
memory (memory cell outputs).

When the **-type** argument is used all the objects in the current instance (or current design, if current instance is not set) that satisfy the selection criteria will be annotated with the specified switching activity. If the **registers** or **regs_on_clocks** selection type is used, both the non-inverting (Q) and inverting (QN) sequential cell outputs are annotated. The non-inverting outputs are annotated with the specified toggle rate and static probability. The inverting outputs are annotated with the specified toggle rate. The annotated static probability on the inverting outputs is $1.0 - sp\_val$ where *sp_val* is the specified static probability. The **regs_on_clocks** selection type needs the argument *clock_list* which is a list of clock names. This selection type selects all the

registers that are clocked by a clock whose name is in *clock_list*. A register is clocked by a clock *clk* if the clock pin of the register is in the transitive fanout of the source port or pin of the clock *clk*. Note that although the **registers** selection type applies to both flip-flops and latches, the **regs_on_clock** selection type applies only to flip-flops. Both the **registers** and **regs_on_clocks** selection types exclude clock-gating cells, that might have internal latches and are therefore sequential cells.

Note that the *object_list* argument for explicitly specifying the objects to be annotated, and the **-type** argument for implicitly specifying the objects to be annotated, are mutually exclusive.

-hierarchy

Used with the **-type** argument and specifies that all the objects in all the hierarchy that satisfy the selection criteria will be annotated. When this option is not specified, only the top level objects in the current instance are considered.

object_list

Specifies a list of nets, pins, ports or cells in the current design on which the **-static_probability** and **-toggle_rate** switching activity values are to be set.

Note that the *object_list* argument for explicitly specifying the objects to be annotated.

If the object is a hierarchical cell, it is used with the **-type** argument and specifies that all the objects in the hierarchical cell that satisfy the selection criteria will be annotated.

-verbose

Provides a more verbose output when switching activity is annotated.

DESCRIPTION

This command can be used to annotate design nets, ports, pins and cells with the different kinds of switching activity. These include simple toggle rate and static probability on nets, ports and pins; state and path dependent toggle rates on cell pins, and state dependent static probabilities on cells.

Toggle rates and static probabilities are annotated using the **-toggle_rate** and **-static_probability** arguments respectively. The toggle rate and static probability can be made state dependent by specifying the state condition with the **-state_dep** argument. The toggle rate can be made path dependent by specifying the path sources with the **-path_sources** argument. A related clock can be specified using the **-base_clock** argument. The annotated toggle rate is divided by the related clock period during power calculation to calculate the effective toggle rate. When none of the **-toggle_rate**, **-static_probability** and **-base_clock** arguments are given, then annotated switching activity is removed from the specified objects.

The design objects that will be annotated with switching activity can be specified explicitly and implicitly as a list of objects. The objects can also be specified implicitly by the **-type** and **-hierarchy** arguments.

For statistics on the switching activity annotation on the current design, use the **report_saif** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following **set_switching_activity** command annotates a simple toggle rate value of $33 / 1320 = 0.025$ and a static probability

value of 0.015 to all design input ports.

```
prompt> set_switching_activity -toggle_rate 33 -period 1320 \
-static_probability 0.015 [get_inputs]
```

The following example annotates a toggle rate value of 0.25 and a static probability value of 0.015 to all design input ports. The clock CLK is specified as the related clock for all inputs. When calculating the power, the annotated toggle rate value (0.25) is divided by the related clock period (10) and the resulting toggle rate (0.025) is used.

```
prompt> create_clock CLK -period 10
prompt> set_switching_activity -toggle_rate 0.25 -base_clock CLK \
-static_probability 0.015 -type inputs
```

The following example annotates state-dependent static probabilities on the cell or1:

```
prompt> set_switching_activity -static_probability 0.10 \
-state_condition "A & B" [get_cell or1]
prompt> set_switching_activity -static_probability 0.35 \
-state_condition "A & ! B" [get_cell or1]
prompt> set_switching_activity -static_probability 0.25 \
-state_condition "! A & B" [get_cell or1]
prompt> set_switching_activity -static_probability 0.30 \
-state_condition "! A & ! B" [get_cell or1]
```

The following example annotates simple and path-dependent toggle rates on the Y output pin of the cell xor1:

```
prompt> set_switching_activity -toggle_rate 0.022 \
[get_pin xor1/Y]
prompt> set_switching_activity -toggle_rate 0.020 \
-path "A" [get_pin xor1/Y]
prompt> set_switching_activity -toggle_rate 0.002 \
-path "B" [get_pin xor1/Y]
```

The following example removes the switching annotation from all sequential cell outputs in the current design:

```
prompt> set_switching_activity -type registers -hierarchy
```

The following example annotates switching activity on all the registers in the current design that are clocked by the clk1 or clk2 clocks.

```
prompt> set_switching_activity -toggle_rate 0.1 \
-static_probability 0.5 \
-type {regs_on_clock {clk1 clk2}} -hierarchy
```

The following example annotates all primary inputs with a toggle rate of 0.2, a static probability of 0.5, and a related_clock attribute with a value of "". When power is calculated during the **report_power** command, the related clocks on the design inputs are inferred automatically and the toggle rate value of 0.2 is scaled by the related clock period.

```
prompt> set_switching_activity -toggle_rate 0.2 \
-static_probability 0.5 -base_clock "" \
-type inputs
prompt> report_power
```

SEE ALSO

reset_switching_activity(2)


```
report_saif(2)  
report_power(2)  
create_clock(2)
```

set_switching_activity_profile

Generates switching activity profiles and sets the profiles on input-bused ports of the current design.

SYNTAX

```
status set_switching_activity_profile
  [-uniform {static_probability toggle_rate}]
  [-linear {{breakpoint_0 static_probability_0 toggle_rate_0} {breakpoint_1 static_probability_1 toggle_rate_1}}]
  [-normal_dist {std_dev temp_corr sample_rate is_signed}]
  [-period period_value | -base_clock clock]
  [object_list]
  [-all_buses]
  [-verbose]
```

Data Types

<i>static_probability</i>	float
<i>toggle_rate</i>	float
<i>breakpoint_0</i>	integer
<i>static_probability_0</i>	float
<i>toggle_rate_0</i>	float
<i>breakpoint_1</i>	integer
<i>static_probability_1</i>	float
<i>toggle_rate_1</i>	float
<i>std_dev</i>	float
<i>temp_corr</i>	float
<i>sample_rate</i>	float
<i>is_signed</i>	Boolean
<i>period_value</i>	float
<i>clock</i>	string
<i>object_list</i>	collection

ARGUMENTS

-uniform {*static_probability* *toggle_rate*}

This optional argument sets the same switching activity on all bits of a bus. The static probability is a positive floating-point number between 0.0 and 1.0. It represents the percentage of time that the signal is at logic 1. For example, a static probability value of 0.25 indicates that the signal is at logic 1 for 25 percent of the time. The toggle rate is a positive floating-point number that represents the number of 0->1 and 1->0 transitions that the signal makes during a period of time. The period is, by default, 1 unit and can be specified with the **-period** option. The time unit defined in the library is used as the time unit. Alternatively, a related clock can be annotated using the **-base_clock** option, and the specified toggle rate is relative to the related clock period.

-linear {{*breakpoint_0* *static_probability_0* *toggle_rate_0*} {*breakpoint_1* *static_probability_1* *toggle_rate_1*}}

This optional argument sets a piecewise linear profile on a bus. The `breakpoint_0` is an integer number between the start bit and end bit of the bus. It represents the first bit position of a bus where the `static_probability_0` and `toggle_rate_0` is applied. Similarly with the `breakpoint_1`, it represents the second bit position of a bus where the `static_probability_1` and `toggle_rate_1` is applied. Each bit between `breakpoint_0` and `breakpoint_1` is set to the corresponding static probability and toggle rate that are calculated by the toggle rate values and static probability values on `breakpoint_0` and `breakpoint_1`.

-normal_dist {std_dev temp_corr sample_rate is_signed}

This optional argument generates a piecewise linear profile for the specified signal distribution and sets this profile on a bus. `std_dev` represents the standard deviation of normal distribution. `temp_corr` is the word-level temporal correlation. `sample_rate` is the number of the input value changes during a period. `is_signed` is a Boolean value to indicate whether inputs are signed or unsigned.

-period period_value

This optional argument specifies the time period for which the number of transitions given in the toggle rate occur. The time units for this value are those specified in the main logic library. When this argument is used, the toggle rate specified is divided by the given "period_value"; the resulting toggle rate is then annotated to the objects specified with the **set_switching_activity_profile** command. If the **-period** argument is not specified, a default of 1.0 is assumed.

-base_clock clock

This optional argument specifies a clock to which the toggle rate is related. This clock is referred to as the object's related clock. When specified, the related clock is annotated to the design objects, and during power calculations the annotated toggle rate value is divided by the related clock period. The resulting toggle rate is then used for calculating power. The clock period can be changed between different runs of "report_power" and the resulting change in the toggle rate of the design objects is used automatically without re-annotating the switching activity. You can also specify "" as the clock value of the **-base_clock** argument. The "" specifies that Power Compiler infers the related clock on the design object automatically before power calculations.

object_list

This optional argument specifies a list of input based ports in the current design on which the switching activity profiles are to be applied. Note that this command only supports in/inout based port. It does not support scalar port types (net, pin and so on). In addition, this command supports partial selection of a bus and a hierarchical bus as an object. The partial selection of a bus is a portion of a full bus. The hierarchical bus indicates a bus not in top-level design, but in a subdesign. The partial selection of a bus string should follow the format `bus_name[start_bit:end_bit]`. The hierarchical bus string should follow the format `cell_name/bus_name`. This command also supports use of the wildcard character in the `object_list`.

-all_buses

This optional argument sets the switching activity profile on all input buses of the current design. Note that **-all_buses** and `object_list` are exclusive options for explicitly specifying the bus to be annotated. Otherwise an error message is issued.

-verbose

This optional argument reports the switching activity profiles that are annotated on the object list.

DESCRIPTION

This command can be used to annotate design input based ports with different kinds of switching activity profiles, including uniform profiles and piecewise linear profiles.

The **-uniform** option can set the same activity on all bits of a bus. The **-linear** option can set a piecewise linear profile for a bus. The **-normal** option can set a piecewise linear profile on a bus that is calculated from the specified signal distribution.

The design objects that are annotated with the switching activity profiles can be specified explicitly as a list of objects. The objects can also be specified by the **-all_buses** flag option.

For statistics on the switching activity annotation on the current design, use the **report_saif** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following **set_switching_activity_profile** command annotates a simple toggle rate value of $33 / 1320 = 0.025$ and a static probability value of 0.015 to an input bus a. The width of bus a is 8.

```
prompt> set_switching_activity_profile -uniform {0.015 33} \
      -period 1320 a -verbose
```

```
*****
Report : Attribute
Design : mult
Version: G-2012.06-BETA3
Date   : Sun Apr 15 21:28:29 2012
*****
```

Bus	Net	Static Probability	Toggle Rate	Bit Position
a	a[0]	0.015000	0.025000	least significant
	a[1]	0.015000	0.025000	
	a[2]	0.015000	0.025000	
	a[3]	0.015000	0.025000	
	a[4]	0.015000	0.025000	
	a[5]	0.015000	0.025000	
	a[6]	0.015000	0.025000	
	a[7]	0.015000	0.025000	most significant

The following **set_switching_activity_profile** command annotates a piecewise linear profile to an input bus a. The width of bus a is 8.

```
prompt> set_switching_activity_profile -linear {{2 0.3 1} {6 0.5 2}} \
      a -verbose
```

```
*****
Report : Attribute
Design : mult
Version: G-2012.06-BETA3
Date   : Sun Apr 15 21:16:32 2012
*****
```

Bus	Net	Static Probability	Toggle Rate	Bit Position
a	a[0]	0.300000	1.000000	least significant
	a[1]	0.300000	1.000000	

a[2]	0.300000	1.000000	breakpoint_0
a[3]	0.350000	1.250000	
a[4]	0.400000	1.500000	
a[5]	0.450000	1.750000	breakpoint_1
a[6]	0.500000	2.000000	
a[7]	0.500000	2.000000	

The following **set_switching_activity_profile** command annotates a normal distribution profile to an input bus a. The width of bus a is 8.

```
prompt> set_switching_activity_profile -normal {4 0 5 0} a -verbose
```

```
*****
```

Report : Attribute

Design : mult

Version: G-2012.06-BETA3

Date : Sun Apr 15 21:24:24 2012

```
*****
```

Bus	Net	Static Probability	Toggle Rate	Bit Position
a	a[0]	0.500000	5.000000	least significant
	a[1]	0.500000	5.000000	
	a[2]	0.500000	5.000000	breakpoint_0
	a[3]	0.333333	3.333333	
	a[4]	0.166667	1.666667	
	a[5]	0.000000	0.000000	breakpoint_1
	a[6]	0.000000	0.000000	
	a[7]	0.000000	0.000000	most significant

SEE ALSO

```
set_switching_activity(2)
reset_switching_activity(2)
report_power(2)
create_clock(2)
```

set_synlib_dont_get_license

Specifies a list of synthetic library part licenses that are not automatically checked out.

SYNTAX

```
int set_synlib_dont_get_license  
    license_list
```

ARGUMENTS

license_list

Lists synthetic library part licenses that are not automatically checked out.

DESCRIPTION

Specifies a list of synthetic library part licenses that are not automatically checked out. By default, all synthetic library part licenses are automatically checked out if there is a possibility that they will be used. Add licenses to this list if they should not be automatically checked out when only the possibility of their use exists. The exception that licenses on this list can be checked out is, but only when read a design that **required** to have these licenses, or when manually requested via the **get_license** command. To determine what licenses are required for reading a design, use **report_design** command.

If all licenses in the list present in the key file, the command sets synlib variable **synlib_dont_get_license** to be the same list of licenses. To determine the current value of **synlib_dont_get_license**, type **list synlib_dont_get_license**.

The **set_synlib_dont_get_license** command fails if any of the license key is mistyped in the command or not present in the key file.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example issues an error message because the license key is mistyped in the command or not present in the key file.

```
prompt> set_synlib_dont_get_license \  
{"Synlib_advMath" "Synlib_ALU"}  
Warning: Unrecognized key 'Synlib_advMath' found in 'synlib_set_dont_get_license' command. (UISN-30)
```

Warning: Unrecognized key 'Synlib_ALU' found in 'synlib_set_dont_get_license' command. (UISN-30)

0

The following example specifies that licenses named DesignWare-Foundation-Ultra and SynLib-AdvMath are not to be automatically checked out.

```
prompt> set_synlib_dont_get_license \  
      {DesignWare-Foundation-Ultra SynLib-AdvMath}  
{DesignWare-Foundation-Ultra SynLib-AdvMath}  
1
```

SEE ALSO

get_license(2)
report_design(2)
synlib_dont_get_license(3)

set_tap_elements

Specifies the set of TAP state elements for the boundary-scan TAP controller.

SYNTAX

```
int set_tap_elements  
  -state_cells cell_names
```

Data Types

cell_names list

ARGUMENTS

-state_cells *cell_names*

Specifies the set of TAP state elements *cell_names*.

DESCRIPTION

The **set_tap_elements** command identifies the set of TAP state elements in the boundary-scan TAP controller. The state elements can be specified in any order. A minimum of four state elements are required according to the IEEE 1149.1 standard.

EXAMPLES

The following example specifies four TAP state elements.

```
prompt> set_tap_elements -state_cells {  
      tap/simple/tap_state_reg[1],  
      tap/simple/tap_state_reg[3],  
      tap/simple/tap_state_reg[0],  
      tap/simple/tap_state_reg[2] }
```

set_target_library_subset

Restricts the optimization of a block to a subset of the target libraries.

SYNTAX

```
status set_target_library_subset
  [-top | -object_list cells]
  [-dont_use lib_cells]
  [-use lib_cells]
  [-clock_path]
  [-only_here lib_cells]
  [-milkyway_reflibs milkyway_reflib_paths]
  [library_file_name_list]
```

Data Types

<i>cells</i>	collection
<i>lib_cells</i>	collection
<i>milkyway_reflib_paths</i>	list
<i>library_file_name_list</i>	list

ARGUMENTS

-top

Sets the target library subset on the top-level design, including all cells in the hierarchy.

If you do not specify either this option or the **-object_list** option, the tool uses the **-top** option by default. Specifying both options is also supported.

-object_list *cells*

Specifies the hierarchical cells on which to set the target library subset.

The cells are instances of hierarchical designs, and the subset restriction applies to these instances and their child instances. You cannot specify a target library subset on leaf-level cells.

If you do not specify either this option or the **-object_list** option, the tool uses the **-top** option by default. Specifying both options is also supported.

-dont_use *lib_cells*

Specifies the library cells that cannot be used for optimization within the block. This is in addition to the restrictions imposed by the *library_file_name_list* argument.

You can specify the library cell by name, such as AN2 or mylib/AN2, or by collections, such as [get_lib_cells */AN2]. You can also

use wildcards. Wildcards are expanded according to the library cells in memory at the time the command is issued. In all cases, the library name is ignored. For example, the **-dont_use "lib1/AN2"** option restricts the use of all library cells that have the name AN2.

-use *lib_cells*

Specifies library cells that can be used for optimization within the block. This is in addition to the libraries listed in the *library_file_name_list* argument.

The syntax for the **-use** and **-dont_use** options are identical. If a library cell is listed in both the **-use** and **-dont_use** options, the **-use** option overrides the **-dont_use** option. A library cell with the **dont_use** attribute is not used even if it is listed in the **-use** option. To remove the **dont_use** attribute, use the **remove_attribute** command. Otherwise, the tool issues a reminder.

If you specify this option and you do not specify the *library_file_name_list* argument, the *library_file_name_list* argument takes on an empty default value.

-clock_path

When this option is specified, the requested target library subset will apply to cells along any clock path within the specified hierarchies, rather than all cells in these hierarchies. These clock path cells will be mapped using library cells in the subset, but will not be optimized further by Design Compiler.

-only_here *lib_cells*

Specifies the library cells that can be used for optimization within the specified blocks but cannot be used in other blocks (unless those blocks also list the library cell in an **-only_here** option).

The syntax for the **-only_here** and **-dont_use** options are identical. If a library cell is listed in both the **-only_here** and **-dont_use** options, the **-only_here** option overrides the **-dont_use** option. However, the **-only_here** option does not override the global **dont_use** attribute set by the **set_dont_use** command. Therefore, a library cell with the **dont_use** attribute is not used even if it is listed by the **-only_here** option.

The target library subset report does not show the effects of the global **dont_use** attribute setting. Therefore, you should check for any LIBSETUP-124 messages when you use the **set_target_library_subset** command. To remove a **dont_use** attribute that was set by the **set_dont_use** command, use the **remove_attribute** command. Otherwise, the tool issues a reminder.

-milkyway_reflibs *milkyway_reflib_paths*

Specifies the Milkyway reference library paths to associate with the identified instances. When you specify multiple reference library paths, enclose them in braces ({}). The list must be a subset of the reference library list for the design.

Design Compiler parses but ignores this option and passes it to the .ddc file or Milkyway database and the **write_script** command.

library_file_name_list

Specifies the libraries that are available to optimize the specified design instances. The list of libraries must be a subset of the libraries specified by the **target_library** variable.

If you do not specify this option, the behavior depends on whether you specify the **-use** option. If you do not specify the **-use** option, all the libraries listed in the **target_library** variable can be used. If you specify the **-use** option, only the cells specified in the **-use** option can be used.

DESCRIPTION

The **set_target_library_subset** command specifies a subset of target libraries that are available for use during the optimization of the specified instances. Usually, a library cell is selected from the libraries specified in the **target_library** variable. The

set_target_library_subset command allows you to specify or filter the library cells to be used in a particular block.

The subset applies to the specified blocks and their child instances. A subset on a low-level block completely overrides a subset specified at a higher level.

The subset restriction applies only to cells that are created, mapped, or modified during optimization. The subset does not affect any unoptimized portions of the design. Use the **check_target_library_subset -verbose** command to see any cells that are already mapped to the library cells outside their subset.

To remove a target library subset from the design or a design instance, use the **remove_target_library_subset** or **reset_design** command.

As with most design constraints, the subset specifications are tied to the current design. If you change the current design to another level, and you want the subset to affect optimization at that level, you must reapply the **set_target_library_subset** command, or use the **characterize** or **propagate_constraints** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the target libraries for the design to lib1.db and lib2.db but restricts the library cells used in block u1 to the lib2.db library.

```
prompt> set_target_library "lib1.db lib2.db"
prompt> set_target_library_subset "lib2.db" -object_list [get_cells u1]
```

The following example restricts the AVOID1 and AVOID2 library cells from use in the u1 and u2 blocks.

```
prompt> set_target_library_subset -object_list "u1 u2" \
-dont_use "AVOID1 AVOID2"
```

The following example restricts the SPECIAL library cell for use only in the u1 and u2 blocks.

```
prompt> set_target_library_subset -object_list "u1 u2" \
-only_here "SPECIAL"
```

The following example has the same result as the previous example. The second **set_target_library_subset** command imposes no restrictions, so the cells from all the libraries specified by the **target_library** variable are allowed in the u1 and u2 blocks, including the SPECIAL library cell.

```
prompt> set_target_library_subset -top -dont_use "SPECIAL"
prompt> set_target_library_subset -object_list "u1 u2"
```

SEE ALSO

remove_target_library_subset(2)
report_target_library_subset(2)
check_target_library_subset(2)
set_operating_conditions(2)
target_library(3)

set_technology

Applies technology-node settings to the current design

SYNTAX

```
status set_technology  
-node node_number
```

Data Types

node_number string

ARGUMENTS

-node *node_number*

Specifies the technology-node settings, such as "7", "12" or "16". You have to specify one of the supported technology node names.

DESCRIPTION

This command applies technology-node settings to the current design. It sets the parameters for placement and route engines to satisfy advanced node design rules. When you use the command, all the technology-node specific controls are applied to the design and the "technology_node" attribute is also set on the design.

It is recommended to use the **set_technology** command at the beginning of the design flow. For example, after you load and link the design, and before place and route.

The settings are persistent and saved in the design database, so you do not have to call this command multiple times in the flow. However, when you move to a newer release, you must use the **set_technology** command again, as the internal settings might get upgraded in the newer release.

You can use this feature in Design Compiler NXT topographical mode.

EXAMPLES

In the following example, the technology-node settings are applied to a design using the 7 nm technology.

```
prompt> set_technology -node 7  
7  
1
```

SEE ALSO

`set_hpc_options(2)`

set_test_assume

Sets the **test_assume** attribute to a logic value to be assumed on specified cell output pins throughout test design rule checking.

SYNTAX

```
integer set_test_assume  
    zero_or_one_value  
    pin_list
```

Data Types

<i>zero_or_one_value</i>	string
<i>pin_list</i>	list

ARGUMENTS

zero_or_one_value

Specifies the fixed value of the output pins. To assume a constant value of logic one, *zero_or_one_value* must be "1", "one", or "ONE." To assume a constant value of logic zero, *zero_or_one_value* must be "0", "zero", or "ZERO."

pin_list

Lists the hierarchical pins in the current design for which values are specified. If more than one pin is specified, they must be enclosed in braces ({ }).

DESCRIPTION

The **set_test_assume** command sets the **test_assume** attribute to *zero_or_one_value* on the cell output pins on *pin_list*. The **dft_drc** command uses **test_assume** to freeze the assumed value on the specified pins throughout test design rule checking.

Use **set_test_assume** to set conditions that would otherwise be unknown to **dft_drc**, such as the state of a non-scan register, or the output of a RAM. Known values are often applied to the design circuitry to facilitate testing by configuring the design circuitry into the desired test mode. For example, during testing, a state could be preloaded into the non-scan registers of an on-chip test controller. This condition could configure the scan chains in the design to form multiple parallel scan chains. Or, if it is known that prior to testing, all address locations of an on-chip RAM are preloaded with a fixed pattern, this pattern can be assumed at the data output pins of the RAM when testing the surrounding functional logic. You can communicate these preloaded values to the testing software through the **test_assume** attribute.

Note: The specified conditions must exist throughout testing, regardless of other changes in the design state. Otherwise, incorrect design rule checking reports can result. If the assumed state is not attainable or not held throughout the scan test sequence, it is

considered best practice to use the initialization protocol to provide initialization vectors that perform the assumed setup, instead of using **set_test_assume**, which could result in errors. The **read_test_protocol-section test_setup** is used to read in this initialization protocol.

Use the **report_test_assume** command to list the **test_assume** attributes on a design.

To remove selected or all **test_assume** attributes from a design, use the **remove_attribute** or **reset_design** command.

EXAMPLES

The following commands specify a fixed internal state to assume for a register in an on-chip test controller:

```
prompt> set_test_assume 0 my_tap/reg3/Q
```

```
prompt> set_test_assume 1 my_tap/reg3/QN
```

The following command specifies that all the data outputs of a RAM cell are fixed throughout testing:

```
prompt> set_test_assume ONE [get_pins h1/ramcell/DOUT*]
```

SEE ALSO

- current_design(2)
- dft_drc(2)
- get_attribute(2)
- remove_attribute(2)
- remove_test_assume(2)
- report_test_assume(2)
- reset_design(2)
- set_dft_signal(2)

set_test_point_configuration

Specifies the automatic test point insertion configuration options (deprecated).

SYNTAX

```
status set_test_point_configuration
  -target pattern_reduction | testability
  [-control_signal control_name]
  [-clock_signal clock_name]
  [-clock_type dominant | dedicated]
  [-max_control_points n]
  [-max_observe_points n]
  [-test_points_per_scan_cell n]
  [-power_saving enable | disable]
  [-max_additional_logic_area n]
```

Data Types

```
control_name  string
clock_name    string
n             integer
```

ARGUMENTS

-target **pattern_reduction** | **testability**

Enables and configures an automatic test point insertion target. This is a required argument. The valid target values are:

- **pattern_reduction** - focuses on pattern count reduction (Test Data Volume Reduction) by inserting only observe test points.
- **testability** - focuses on test coverage improvement by inserting both control and observe points.

The **-target** option specifies which automatic test point insertion target to enable, and it is a required option. The **pattern_reduction** and **testability** targets are mutually exclusive.

-control_signal **control_name**

Specifies the name of the control signal for test points. The signal must have a type of TestMode or IbistEnable. If the control signal is not specified, the test points are controlled by any available signal with the TestMode signal type. If a TestMode signal is not found, DFT Compiler creates a new test-mode signal.

-clock_signal **clock_name**

Specifies the name of the clock signal used for test points when a dedicated test point clock is used. If dedicated test point clocks are enabled with the **-clock_type** option and a clock signal is not specified with this option, DFT Compiler creates a new dedicated test point clock port.

In all flows, you can specify the name of a scan clock signal that is predefined with the **set_dft_signal -type ScanClock** command.

In a DFT-inserted OCC controller flow, you can specify the name of a PLL output pin that is predefined with the **set_dft_signal -type Oscillator** command. This specification is mapped to the corresponding OCC controller clock during DFT insertion.

In an existing OCC controller flow, you can directly specify the name of an output pin of an existing OCC controller that is predefined with the **set_dft_signal -type Oscillator** command.

-clock_type dominant | dedicated

Specifies the clock type to use for control and observe point scan cells. A value of **dominant** clocks the test points with the clock signal that clocks the most sequential cells in the design. A value of **dedicated** uses a separate dedicated test point clock. By default, the clock type is **dominant**.

-max_control_points *n*

Specifies the maximum number of control test points inserted. The default is 1000.

This option can be used only with the **testability** target.

-max_observe_points *n*

Specifies the maximum number of observe test points inserted. The default is 1000.

-test_points_per_scan_cell *n*

Specifies the number of test points of the same type (control or observe test points) that can be shared per scan cell. The valid value range is 1 to 8. By default, eight control test points can share one control scan cell and eight observe test points can share one observe scan cell.

-power_saving enable | disable

Enables the insertion of power-saving logic structures (AND gates) on the observe XOR trees.

-max_additional_logic_area *n*

Specifies the maximum percentage of area overhead allowed by automatic test point insertion.

DESCRIPTION

Note: this command has been deprecated. Use the **set_testability_configuration** command instead.

The **set_test_point_configuration** command configures the automatic test point insertion. When enabled, the design is analyzed during DFT insertion, and then test points are inserted to improve the testability of the design.

This command configures automatic test point insertion, but does not enable it. To enable automatic test point insertion, enable it with the **set_dft_configuration** command, then configure one or more test point targets with this command:

```
set_dft_configuration -test_points enable
set_test_point_configuration -target ...
set_test_point_configuration -target ...
```

You can insert test points at either the RTL or gate level. If you insert observe test points at the RTL level, set the **compile_delete_unloaded_scan_cells** variable to **false** to avoid removing scan cells associated with observe points during compilation.

In order to get the optimum test coverage improvement, both control and observe test points should be inserted by using the **-target**

testability option.

EXAMPLES

The following example inserts up to 6 control and up to 1000 (default) observe test points:

```
prompt> set_test_point_configuration -target testability \  
        -max_control_points 6 \  
        -clock_type dominant -test_points_per_scan_cell 2
```

The following example inserts up to 6 control and up to 10 observe test points:

```
prompt> set_test_point_configuration -target testability \  
        -max_control_points 6 -max_observer_points 10 \  
        -clock_type dominant -test_points_per_scan_cell 2
```

The following example inserts up to 7 observe test points:

```
prompt> set_test_point_configuration -target pattern_reduction \  
        -max_observe_points 7 \  
        -clock_type dedicated -test_points_per_scan_cell 3
```

The following example inserts up to 1000 (default) control and 1000 (default) observe test points:

```
prompt> set_test_point_configuration -target testability \  
        -clock_type dedicated
```

SEE ALSO

report_test_point_configuration(2)
reset_test_point_configuration(2)
set_dft_configuration(2)

set_test_point_element

Configures the insertion of force, control, and observe user-defined test points.

SYNTAX

```
int set_test_point_element
    list_of_design_pin_objects
    [-type control_0 | control_z0 | control_1 | control_z1 | control_01 | control_z01 | force_0 | force_z0 | force_1 | force_z1 | force_01 | for
    [-control_signal control_name]
    [-clock_signal clock_name]
    [-source_or_sink source_or_sink_name]
    [-test_point_enable test_point_enable_name]
    [-test_points_per_source_or_sink n]
    [-test_points_per_test_point_enable n]
    [-scan_source_or_sink enable | disable]
    [-scan_test_point_enable enable | disable]
    [-power_saving enable | disable]
```

Data Types

<i>control_name</i>	string
<i>clock_name</i>	string
<i>source_or_sink_name</i>	string
<i>test_point_enable_name</i>	string
<i>n</i>	integer

ARGUMENTS

-type control_0 | control_z0 | control_1 | control_z1 | control_01 | control_z01 | force_0 | force_z0 | force_1 | force_z1 | force_01 | force_z01 | observe

| control_z1 | control_01 | control_z01 | Oforce_0 | force_z0
| force_1 | force_z1 | force_01 | force_z01 | observe

Specifies the type of test point to insert at each design object in the list. Specify only one type with each invocation of this command. Observe and control test points are not supported with the LSSD scan style.

-control_signal *control_name*

Specifies the name of the existing pin or port control signal that activates the test point. The signal must have a type of TestMode, ScanEnable, or IbiStEnable. If no control signal is specified, the test points are controlled by any signal with the TestMode signal type. If no TestMode signal is found, DFT Compiler creates a new TestMode port.

-clock_signal *clock_name*

Specifies the name of the clock signal used to clock any inserted user-defined test point scan registers. If no clock signal is specified, DFT Compiler creates a new clock port.

In all flows, you can specify the name of a scan clock signal that is predefined with the **set_dft_signal -type ScanClock** command.

In a DFT-inserted OCC controller flow, you can specify the name of a PLL output pin that is predefined with the **set_dft_signal -type Oscillator** command. This specification is mapped to the corresponding OCC controller clock during DFT insertion.

In an existing OCC controller flow, you can directly specify the name of an output pin of an existing OCC controller that is predefined with the **set_dft_signal -type Oscillator** command.

-source_or_sink source_or_sink_name

Specifies the name of the source data signal (for control and force test points) or the sink observed data signal (for observe test points). The source or sink signal can be an existing pin or an existing port. This option overrides the normal creation of source or sink scan registers (when **-scan_source_or_sink** is enabled) or source or sink ports (when **-scan_source_or_sink** is disabled).

When a signal is specified with the **-source_or_sink** option, it will be used for all source or sink signal connections for that test point definition, and the **-test_points_per_source_or_sink** sharing value will be ignored.

-test_point_enable test_point_enable_name

Specifies the name of the test point enable signal (for control test points). The enable signal can be an existing pin or an existing port. This option overrides the normal creation of test point enable scan registers (when **-scan_test_point_enable** is enabled) or test_point_enable input ports (when **-scan_test_point_enable** is disabled).

When a signal is specified with the **-test_point_enable** option, it will be used for all test point enable signal connections for that test point definition, and the **-test_points_per_test_point_enable** sharing value will be ignored.

-test_points_per_source_or_sink n

Specifies the number of test points that can be shared per source data signal (for control and force test points) or sink observed signal (for observe test points). The source signal can be a source data scan register or a source signal input port. The sink signal can be an observe scan cell or a sink output port.

A range of 1 to 32 can be specified for *n*.

By default, 8 observe test points can share one sink signal and 8 control or force test points can share one source signal.

-test_points_per_test_point_enable n

Specifies the number of test points that can be shared per test point enable signal for control test points. By default, a test point enable signal is used per control test point. This test point signal can be a test point enable scan register or a test point enable input port.

This option is valid only for the control test point types.

-scan_source_or_sink enable | disable

Enables or disables the insertion of control or observe scan registers. When **-scan_source_or_sink** is enabled (which is the default behavior), the source data of the force and control test points come from new source signal scan registers, and data is observed through sink scan registers. When **-scan_source_or_sink** is disabled, the source data of the control and force test points come from a new source signal input ports, and data is observed through new sink output ports.

If the **-source_or_sink** option is specified, this option is ignored.

-scan_test_point_enable enable | disable

Enables or disables the insertion of test point enable scan registers. When **-scan_test_point_enable** is enabled (which is the default behavior), the test point enable signal for the control test points come from new test point enable scan registers. When -

scan_test_point_enable is disabled, the test point enable signal for the control test points come from a new test point enable signal input port.

If the **-test_point_enable** option is specified, this option is ignored.

-power_saving enable | disable

Specifies the insertion of power saving logic structures (AND gates) on the observe XOR trees.

DESCRIPTION

The **set_test_point_element** command configures the insertion of force, control, and observe user-defined test points. These test points are subsequently modeled by the **preview_dft** command, and inserted by the **insert_dft** command.

This command requires a list of pins or ports where the test points should be inserted. The user-defined test point feature can fix scan rule violations (force test points), reduce test data volume reduction (observe test points) and improve test coverage (control and observe test points).

Force test points are used when a value needs to be forced throughout the entire test session. A force test point is activated when the control signal (normally the TestMode signal) is asserted. The force_0 and force_1 test point types allow a signal to be replaced with a constant 0 or constant 1 value throughout the entire test session. These test point types are useful when a particular signal must be forced to a known value for testability purposes. The force_01 test point type allows a signal to be replaced with a source data value (driven from a scan register, input port, or user-supplied source data signal) throughout the entire test session.

Control test points are used when a hard-to-control signal should be controllable (selectively forced) for some test vectors, but left unaltered for others. This allows the test program to select either the original signal behavior, or the forced behavior, on a vector-by-vector basis. Control test points are typically inserted to increase the fault coverage of the design. A control test point is like a "selectively-activated" force control point. It works the same way, except that an additional test point enable signal, supplied by a scan register, input port, or user-supplied test point enable signal, must also be asserted.

Observe test points are typically inserted at hard-to-observe signals in a design to reduce test data volume or to increase the coverage. An observe test point passively observes a signal, and captures its value in an observe sink data scan register, or a sink data output port. Multiple observed points within a single observe test point definition can share the same sink point by collapsing the observed signals with an XOR observability tree.

Each **set_test_point_element** command describes a unique test point element definition. Signal sharing is not performed between test point element definitions. If test points within a limited geographic region should share the same source, sink, or enable signals, they should all be provided in a single **set_test_point_element** command. If test points across a wide geographic region should not share signals to avoid routing congestion, they should be broken up into localized groups and specified with separate **set_test_point_element** commands.

EXAMPLES

The following example specifies that observe test points be inserted at the specified set of four design pins. Since **-scan_source_or_sink** is enabled by default, **insert_dft** inserts two new observe scan registers to observe the four pins. The first three pins share one observe scan register, and the last pin is observed by a second observe scan register.

```
prompt> set_test_point_element -type observe {U0/Q U1/Q U2/Q U3/Q} \
      -test_points_per_source_or_sink 3 \
      -clock_signal CLK
```

The following example forces the RSTN pin of an analog block to zero during the entire test program, to hold the analog block in a

quiet, low-power reset state:

```
prompt> set_test_point_element -type force_0 {U_ANALOG/RSTN}
```

SEE ALSO

`insert_dft(2)`
`preview_dft(2)`
`remove_test_point_element(2)`
`report_test_point_element(2)`

set_testability_configuration

Configures automatic test-point insertion and global test-point insertion parameters.

SYNTAX

status **set_testability_configuration**

```
[-target random_resistant | untestable_logic | x_blocking | multicycle_paths | shadow_wrapper | core_wrapper | user]
[-test_point_file file_name]
[-only_from_file false | true]
[-clock_signal clock_name]
[-control_signal control_name]
[-test_points_per_scan_cell n]
[-include_elements object_list]
[-include_fanin_cone object_list]
[-include_fanout_cone object_list]
[-exclude_elements object_list]
[-exclude_fanin_cone object_list]
[-exclude_fanout_cone object_list]
[-effort low | medium | high]
[-isolate_elements cell_list]
[-max_test_points n]
[-random_pattern_count n]
[-target_test_coverage coverage_value]
[-reuse_threshold threshold_value]
[-depth_threshold threshold_value]
[-sg_command_file file_name]
Values: random_resistant | untestable_logic | x_blocking | multicycle_paths | core_wrapper | shadow_wrapper | user)
```

Data Types

```
file_name      string
clock_name     string
control_name  string
n              integer
object_list   list
cell_list     list
coverage_value float
threshold_value integer
```

ARGUMENTS

-target random_resistant | untestable_logic | x_blocking | multicycle_paths | shadow_wrapper | core_wrapper | user

Enables and configures an automatic test point insertion target.

Test points are not inserted for a target until it is enabled by this option. Target lists are supported.

The valid target values are:

- **random_resistant** - inserts **control_0**, **control_1**, and **observe** test points to improve the testability of hard-to-test (random-pattern-resistant) logic.
- **untestable_logic** - inserts **force_01** and **observe** test points to control uncontrollable nets and observe unobservable logic, respectively.
- **x_blocking** - inserts **force_01** test points at unknown-function output pins, such as the output pins of black boxes.
- **multicycle_paths** - inserts capture-blocking logic at endpoints constrained by multicycle paths.
- **shadow_wrapper** - inserts a shadow wrapper around each cell specified with the **-isolate_elements** option (**observe** test points at input pins, **force_01** test points at output pins).
- **core_wrapper** - inserts test-point-based core wrapping in the design (**force_01** test points at input ports, **observe** test points at output ports).
- **user** - inserts user-supplied test points described in the external file defined by the **-test_point_file** option

If the **-target** option is omitted, the command can only configure the following global (non-target-specific) options:

- **-test_point_file**
- **-clock_signal**
- **-control_signal**
- **-test_points_per_scan_cell**
- **-sg_command_file**

-test_point_file *file_name*

Specifies a file that describes test points to insert.

Each line contains three whitespace-separated fields: a user-defined label (can be any string), the test point type keyword, and the net where the test point should be inserted. Any text after the '#' comment character is ignored, whether in-line or on a separate line.

If a test point comes from a SpyGlass run (with an in-line SpyGlass comment), it belongs to that target. Otherwise, it belongs to the **user** target.

Note that file-based test points are not implemented unless the target is enabled (just as with analysis-based test points). For more information, see the DESCRIPTION section.

Supported scopes: global

-only_from_file false | true

Specifies whether file-based test points should augment or replace analysis-based test points.

When set to **false** (the default) for a target, the tool performs analysis for it, then implements both analysis-based and file-based test points for it.

When set to **true** for a target, the tool does not perform analysis for it; it implements only file-based test points for it. Any options that affect analysis for that target will have no effect.

This option is valid only when the **-test_point_file** option is also set. For more information, see the DESCRIPTION section.

Supported scopes: all targets

-clock_signal *clock_name*

Specifies the name of a previously defined ScanClock signal to use for DFT-inserted test point registers.

The default is to use the dominant clock of the logic surrounding each test point.

Supported scopes: global, all targets

-control_signal *control_name*

Specifies the name of control signal to use for enabling test points. The signal must have a type of TestMode or IbistEnable.

The default is to use an available TestMode signal, or if one is not found, to create a new TestMode signal.

Supported scopes: global, all targets

-test_points_per_scan_cell *n*

Specifies the number of test points that can be shared by a single test-point register.

The default is 8.

Supported scopes: global, all targets

-include_elements *object_list*

Specifies hierarchical cells that should be fixed. Wildcards and collections are supported.

The default is to perform fixing across the entire design.

Supported scopes: **random_resistant**, **untestable_logic**

-include_fanin_cone *object_list*

Specifies ports or pins whose fanin logic should be fixed. Wildcards and collections are supported.

The default is to perform fixing across the entire design.

Supported scopes: **random_resistant**, **untestable_logic**

-include_fanout_cone *object_list*

Specifies ports or pins whose fanout logic should be fixed. Wildcards and collections are supported.

The default is to perform fixing across the entire design.

Supported scopes: **random_resistant**, **untestable_logic**

-exclude_elements *object_list*

Specifies objects that should be excluded from fixing. Wildcards and collections are supported.

The default is not to exclude any objects from fixing.

Supported scopes: **random_resistant** (hierarchical cells), **untestable_logic** (hierarchical cells), **core_wrapper** (ports)

-exclude_fanin_cone *object_list*

Specifies ports or pins whose fanin logic should be excluded from fixing. Wildcards and collections are supported.

The default is not to exclude any objects from fixing.

Supported scopes: **random_resistant**, **untestable_logic**

-exclude_fanout_cone *object_list*

Specifies ports or pins whose fanout logic should be excluded from fixing. Wildcards and collections are supported.

The default is not to exclude any objects from fixing.

Supported scopes: **random_resistant**, **untestable_logic**

-effort *low* | *medium* | *high*

Specifies the effort level used to compute the optimal test points for hard-to-test logic. Higher effort levels yield more efficient test points, but at the expense of analysis runtime.

The default is **medium**.

Supported scopes: **random_resistant**

-isolate_elements *cell_list*

Specifies hierarchical, black-box, or CTL-modeled cells to be isolated with a test-point shadow wrapper. Wildcards and collections are supported.

The default is an empty list.

Supported scopes: **shadow_wrapper**

-max_test_points *n*

Specifies the maximum number of test points to be inserted.

This limit applies to the **random_resistant** and **untestable_logic** targets. When applied globally, the tool allocates the limit across both targets. You can also apply per-target limits, which takes precedence.

The default is 5% of the register count of the design.

Supported scopes: global, **random_resistant**, **untestable_logic**

-random_pattern_count *n*

Specifies the number of random patterns to statistically model when performing random-resistant analysis. Coverage values specified with the **-target_test_coverage** option are interpreted according to this value.

The default is 8000.

Supported scopes: **random_resistant**

-target_test_coverage *coverage_value*

Specifies the random-pattern target test coverage in percentage points that, when reached, prevents further test points from being generated.

The default is 100.

Supported scopes: **random_resistant**

-reuse_threshold *threshold_value*

Specifies the maximum number of I/O registers that can be reused as shared wrapper cells when wrapping the design with test points.

The default is 0.

Supported scopes: **core_wrapper**

-depth_threshold *threshold_value*

Specifies the maximum combinational depth of I/O registers that can be reused as shared wrapper cells when wrapping the design with test points.

The default is 1.

Supported scopes: **core_wrapper**

-sg_command_file *file_name*

Specifies a file of SpyGlass Tcl commands file to be applied to the test-point analysis.

This options file is an unordered list of SpyGlass Tcl commands that you want to apply. The tool automatically applies the Tcl commands at the proper point in the analysis based on their type (design read, setup, goal definitions, etc.).

Supported scopes: global

DESCRIPTION

The **set_testability_configuration** command enables and configures automatic test-point insertion for one or more targets.

To use this feature, you must first enable the testability DFT client:

```
prompt> set_dft_configuration -testability enable
```

Issue this command without the **-target** option to configure global test-point insertion parameters. These global options are noted above.

Issue this command with the **-target** option to configure the specified target(s). Each target supports a particular subset of this command's options, described above. Warning messages are issued if you specify unsupported options for a target.

Specify the global configuration first, followed by the per-target configurations.

The following global options can also be specified per-target, which takes precedence over their global value:

- **-clock_signal**
- **-control_signal**
- **-test_points_per_scan_cell**

By default, analysis-based test points are implemented for a target when it is enabled. If you specify the **-test_point_file** option, any file-based test points are also implemented for that target. To suppress analysis and implement only the file-based test points, set the **-only_from_file** option to **true** for that target.

EXAMPLES

The following example configures three automatic test-point insertion targets:

```
prompt> # global settings
prompt> set_testability_configuration \
    -test_points_per_scan_cell 20 \
    -control_signal TM_TP
```

```
prompt> # target-specific settings
prompt> set_testability_configuration -target random_resistant \
-random_pattern_count 1000 -target_test_coverage 95

prompt> set_testability_configuration -target {x_blocking core_wrapper}
```

The following example configures insertion of only file-based test points for two targets from a previous SpyGlass run, without rerunning analysis:

```
prompt> # global settings
prompt> set_testability_configuration \
-control_signal TM_TP \
-test_point_file spyglass.txt

prompt> # enable targets for file-based insertion only
prompt> set_testability_configuration -only_from_file true \
-target {random_resistant x_blocking}
```

SEE ALSO

```
report_testability_configuration(2)
reset_testability_configuration(2)
run_test_point_analysis(2)
set_dft_configuration(2)
```

set_timing_derate

Sets a derating factor on the current design or specified objects. Derating factors adjust the worst-case calculated delays for a particular operating condition.

SYNTAX

```
status set_timing_derate
  [-rise]
  [-fall]
  [-min]
  [-max]
  [-early]
  [-late]
  [-clock]
  [-data]
  [-net_delay]
  [-cell_delay]
  [-cell_check]
  [-pocvm_guardband]
  [-pocvm_coefficient_scale_factor]
  value
  object_list
```

Data Types

value float
object_list list

ARGUMENTS

-rise

Applies the derating value only to the delays of paths that have a rising transition at the specified objects.

-fall

Applies the derating value only to the delays of paths that have a falling transition at the specified objects.

If neither the **-rise** nor **-fall** option is used in the command, the derating value applies to both rise and fall.

-min

Applies the derating value only to the minimum operating condition.

-max

Applies the derating value only to the maximum operating condition.

If neither the **-min** nor **-max** option is used in the command, the derating value applies to both operating conditions.

-early

Applies the derating factor only to minimum delays, which defines how early a signal can arrive. The factor applies to clock path delays in a setup check, and to data path delays in a hold check.

-late

Applies the derating factor only to maximum delays, which defines how late a signal can arrive. The factor applies to data path delays in a setup check, and to clock path delays in a hold check.

If neither the **-early** nor **-late** option is used in the command, the derating value applies to early and late delays.

-clock

Applies the derating value only to elements in the clock network.

-data

Applies the derating value only to elements in the data network.

If neither the **-clock** nor **-data** option is used in the command, the derating value applies to both clock paths and data paths.

-net_delay

Applies the derating value only to net delay, which is the delay from the driver to the receiver of each net. This option cannot be used with an object list. There is only one set of derating values for net delays and they apply to the whole design.

-cell_delay

Applies the derating value only to cell delay, which is the delay from the input to the output of a cell.

-cell_check

Applies the derating value to the setup and hold time requirements of cells: cell setup and cell recovery times for late derating or cell hold and cell removal times for early derating. This option cannot be specified with any of the following options: **-clock**, **-data**, **-cell_delay**, **-net_delay**, **-aocvm_guardband** and **-pocvm_guardband**.

-pocvm_guardband

This option applies only to the parametric on-chip variation (POCV) context. In that context, both the POCV coefficient and the distance-based derating factor (if used) are adjusted by the *value* specified in the command. This option cannot be used together with the **-dynamic** or **-cell_check** option.

-pocvm_coefficient_scale_factor

This option applies only to the parametric on-chip variation (POCV) context. In that context, the POCV coefficient is adjusted by the *value* specified in the command. For example, if the POCV coefficient is 0.05 and the *value* is 1.2, then the coefficient used for POCV analysis is $0.05 \times 1.2 = 0.06$.

value

Specifies the derating value, which must be in the range of 0.1 to 2.0.

object_list

Specifies the leaf cells, instances, or library cells on which to set the derating factor. To ensure that the setting on an instance is preserved during optimization, the tool sets the **size_only** attribute of the instance to **true**.

DESCRIPTION

The **set_timing_derate** command sets derating factors for the current design or specified objects. If a timing derating factor is specified, some or all path delays are multiplied by the derating factor.

Derating factors define the lower and upper ranges of timing delays for a certain operating condition. The derating factor specified with the **-late** option is multiplied by the data path delays for maximum delay (setup) checking and clock path delays for minimum delay (hold) checking. Conversely, the derating factor specified with the **-early** option is multiplied by the data path delays for hold checking and clock path delays for setup checking.

The **-net_delay**, **-cell_delay**, and **-cell_check** options apply the specified derating value to net delays, cell delays, or cell setup and hold time requirements, respectively. The **-net_delay** option can apply to the whole design only, so it cannot be used with an object list.

If none of these three options is used and an object list is provided, the derating value applies to cell delay only; if no object list is provided, the value applies to both cell delay and net delay.

The setup timing check is calculated using the maximum operating condition. The hold timing check is calculated using the minimum operating condition.

Input delay and ideal clock network latency are not derated.

Multicorner-Multimode Support

This command applies to the current scenario only. By default, when you set timing derating factors on a library cell, the setting is applied to all scenarios. You can control whether timing derating factors on library cells are scenario-specific with the **timing_library_derate_is_scenario_specific** variable. If this variable is **true** when you set the timing derating factor on a library cell, the timing derating factors apply to the current scenario only. For more information, see the man page for the **timing_library_derate_is_scenario_specific** variable.

EXAMPLES

The following example specifies an early derating value of 0.9 and a late derating value of 1.1 for all clock network nets at the maximum operating condition.

```
prompt> set_timing_derate -max -early -net_delay 0.9
prompt> set_timing_derate -max -late -net_delay 1.1
```

The following example specifies an early derating value of 0.8 and a late derating value of 1.2 for leaf cell U1 at the maximum operating condition.

```
prompt> set_timing_derate -max -early 0.8 U1
prompt> set_timing_derate -max -late 1.2 U1
```

The following example specifies an early POCV guardband derating value of 0.8 and a late POCV guard-band derating value of 1.2 for leaf cell U1.

```
prompt> set_timing_derate -pocvm_guardband -early 0.8 U1
prompt> set_timing_derate -pocvm_guardband -late 1.2 U1
```


SEE ALSO

report_timing_derate(2)
report_timing(2)

set_timing_ranges

Sets timing ranges for the current design.

SYNTAX

```
int set_timing_ranges  
    [timing_ranges]  
    [-library library_name]
```

Data Types

<i>timing_ranges</i>	collection
<i>library_name</i>	string

ARGUMENTS

timing_ranges

One or two timing ranges to be applied to the current design.

-library *library_name*

Specifies the name of the library that contains definitions of the timing ranges used.

DESCRIPTION

Specifies the names of one or two timing ranges to be used for timing the current design. The specified timing ranges must be defined in *library_name*, or in one of the libraries in the **link_library**. If a **local_link_library** is set on the current design, it is added to the beginning of the **link_library** before the **link_library** is searched. That is, the order for a library search is:

1. *library_name*
2. **local_link_library**
3. **link_library**

Timing ranges define scaling factors that are used to scale timing path totals. You can model operating condition variations using timing ranges, by defining a range of relative slow and fast times.

The *slowest_factor* is the largest value of any of the specified timing ranges. The *fastest_factor* is the smallest value of any of the specified timing ranges. Maximum delays of data paths are scaled by the *slowest_factor*. Minimum delays of data paths are scaled by the *fastest_factor*. Timing ranges do not affect the clock network. The effect of this command can be seen with the **report_timing**,

report_constraint, and **compile** commands. Timing ranges affect path delays for reports, such as **report_timing** and **report_constraints**, and affect delay cost computation during optimization.

The following is an example of a timing range definition, as it appears in the source text of a library:

```
timing_range("BEST_CASE") {
  faster_factor : 0.95
  slower_factor : 0.99
}
```

The name of this timing range is "BEST_CASE". The fields are defined as follows:

faster_factor

A floating-point number by which arrival times are scaled to model faster times due to environmental variations.

slower_factor

A floating-point number by which arrival times are scaled to model slower times due to environmental variations.

The **report_lib -timing_arcs** command shows the timing ranges that are defined in the technology library.

Each time the **set_timing_ranges** command is used, it overwrites the previous specifications. To remove the timing ranges defined for the current design, use the **set_timing_ranges** command with no parameters, or use the **reset_design** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

To check a library for timing ranges, use the following command:

```
prompt> report_lib test_lib
.
.
.
Timing ranges:
Name      Library  slower_factor  faster_factor
-----
BEST_CASE test_lib  0.98          0.96
WORST_CAS test_lib  1.1           1.0
```

In the following example, timing ranges from "my_lib.db" for the current design are selected:

```
prompt> read my_lib.db
prompt> set_timing_ranges WORST_CASE -library my_lib.db
```

The following example specifies two timing ranges for the current design if the **link_library** is "other_lib.db" and there is no **local_link_library** set on the current design:

```
prompt> set_timing_ranges {WORST_CASE BEST_CASE}
```

SEE ALSO

- compile(2)
- current_design(2)
- report_constraint(2)
- report_lib(2)
- report_timing(2)
- reset_design(2)
- set_local_link_library(2)

set_tlu_plus_files

Sets the files used for TLUPlus extraction. This command is supported only in topographical mode.

SYNTAX

```
status set_tlu_plus_files
  [-max_tluplus max_tluplus_file]
  [-min_tluplus min_tluplus_file]
  [-max_emulation_tluplus max_emul_file]
  [-min_emulation_tluplus min_emul_file]
  [-tech2itf_map mapping_file]
```

Data Types

<i>max_tluplus_file</i>	string
<i>min_tluplus_file</i>	string
<i>max_emul_file</i>	string
<i>min_emul_file</i>	string
<i>mapping_file</i>	string

ARGUMENTS

-max_tluplus *max_tluplus_file*

Specifies the name of the TLUPlus file used for maximum condition calculation of resistance and capacitance using TLUPlus.

-min_tluplus *min_tluplus_file*

Specifies the name of the TLUPlus file used for minimum condition calculation of resistance and capacitance using TLUPlus.

-max_emulation_tluplus *max_emul_file*

Specifies the name of the emulation TLUPlus file used for maximum condition calculation of resistance and capacitance using TLUPlus.

-min_emulation_tluplus *min_emul_file*

Specifies the name of the emulation TLUPlus file used for minimum condition calculation of resistance and capacitance using TLUPlus.

-tech2itf_map *mapping_file*

Specifies the name of the file that defines the layer mapping between the technology file and the Interconnect Technology Format (ITF) file. This switch is optional only if the names are the same.

DESCRIPTION

This command specifies the files used for virtual route and postroute extraction using TLUPlus. At a minimum, you must specify the maximum TLUPlus file and the mapping file.

The **-min_tluplus** option is optional and, if specified, the command runs an extraction for the minimum condition, as well.

The file specified by the **-tech2itf_map** option maps the layer names in the technology file to the layer names in the Interconnect Technology Format (ITF) file. The mapping file should contain at least two sections, `conducting_layers` and `via_layers`; the `conducting_layers` section maps a routing layer to a conductor layer in the ITF file and the `via_layers` section maps a contact layer to a via layer in the ITF file. If the layer names are the same in your technology file and ITF file, you do not have to specify this option. If you do not specify this option, the tool assumes that the layer names are the same. Thus, if the names are actually different, an error will occur later in the flow.

You can also use this command to define two (optional) emulation TLUPlus files, which are used to perform emulation metal fill extraction. Emulation TLUPlus files include metal-fill-related information, such as `FILL_RATIO`, `FILL_SPACING`, `FILL_WIDTH`, and `FILL_TYPE`.

If you use the same names in different directories for the maximum and minimum TLUPlus files, they are considered the same file. The file names must be different if the files are not the same.

Note that TLUPlus files are stored as physical attributes of the design. When you use the **import_designs** command, you must set the TLUPlus files after running this command. Designs without a floorplan are considered logical designs. When a logical design is opened, only logical attributes are reloaded. It is important to run the **initialize_floorplan** or **read_def** command with valid floorplan DEF data after you set the TLUPlus files and before you save the design, to ensure that the physical attributes are reloaded when the design is reopened.

You can read `nxtgrd` files for extraction as an alternative to TLUPlus files. You specify an `nxtgrd` file the same way you specify a TLUPlus file, by using the **set_tlu_plus_files** command as shown in the following example. In this example, the command sets the `NXTGRD_file_max.nxtgrd` file for the maximum condition calculation of the resistance and capacitance:

```
prompt> set_tlu_plus_files -max_tluplus NXTGRD_file_max.nxtgrd -tech2itf_map design.map
```

You must generate the `nxtgrd` file using the `grdgenxo` tool version N-2017.12 or later.

The DC-Synth-AdvGeo license is required when you use the **set_tlu_plus_files** command to read `nxtgrd` files for 5-nm and smaller technology nodes. If the license is not available, the tool issues an error message.

Multicorner-Multimode Support

This command uses information from the current scenario only. If using emulation, it must be set for all scenarios.

EXAMPLES

Assume that the technology file has the following layer definitions:

```
Layer "METAL1" {
  maskName = "metal1"
  ...
}
Layer "VIA1" {
  maskName = "via1"
  ...
}
```

```
}
```

and the Interconnect Technology Format (ITF) file has the following layer definitions:

```
CONDUCTOR metal1 {THICKNESS=0.53 WMIN=0.23 SMIN=0.23 RPSQ=0.078 \  
  CAPACITIVE_ONLY_ETCH=0}
```

```
...
```

```
VIA via1 {FROM=metal1 TO=metal2 AREA=0.0169 RPV=1.6}
```

In this case, the mapping file would have the following entries:

```
conducting_layers
```

```
METAL1 metal1
```

```
...
```

```
via_layers
```

```
VIA1 via1
```

```
...
```

The following example runs the **set_tlu_plus_files** command.

```
prompt> set_tlu_plus_files -max_tluplus tlu.max \  
-tech2itf_map design.map
```

SEE ALSO

`extract_rc(2)`

`set_extraction_options(2)`

set_top_implementation_options

Sets the top-level design options for the linking and optimization of the block's interface logic using transparent interface optimization.

SYNTAX

```
status set_top_implementation_options
  -block_references reference_names
  [-load_logic full_interface | compact_interface]
  [-optimize_block_interface true | false]
  [-optimize_shared_logic true | false]
  [-size_only_mode off | auto | in_place | footprint]
  [-block_update_setup_script file_name]
  [-block_update_cmd_script file_name]
  [-reset]
```

Data Types

```
reference_names    list
file_name          string
```

ARGUMENTS

-block_references *reference_names*

Specifies a list of names of the block references that are to be linked with block abstractions. The tool searches the current design library and its reference libraries for the blocks. The tool also searches the link libraries for blocks created in Design Compiler.

This is a required option.

-load_logic full_interface | compact_interface

Specifies the extent of logic to be loaded from the blocks at the top level. The block must already have a block abstraction created by using the **create_block_abstraction** command.

You can control the extent of the logic to load using one of the following settings:

full_interface (the default)

To load the entire block abstraction, consisting of all the interface logic, use the **-load_logic full_interface** option setting. When you load the full interface, the following logic is marked as interface logic:

- All the logic from input ports to registers or output ports
- All the logic to output ports from registers or input ports

compact_interface

To load a compact version of the block abstraction, consisting of only the logic for the timing-critical paths of every input and output, use the **-load_logic compact_interface** option setting.

For compact block abstractions, the following logic is marked as interface logic:

- The logic that belongs to min/max rise/fall from input ports to registers or output ports
- The logic that belongs to min/max rise/fall to output ports from registers or input ports

The following clock-gating circuitry is marked as interface logic whether you load the full interface or the compact interface:

- Any logic in the connection from the master clock to the generated clocks
- The clock trees that drive interface registers, including any logic in the clock tree
- The longest and shortest clock paths from the clock ports
- The side-load cells of all non-ideal and non-DRC disabled nets

When transparent interface optimization is not performed on a block, you can load the block with a compact interface. However, when you perform transparent interface optimization at the top level with the **compile_ultra** command, load the entire block abstraction for the blocks you want to optimize by using the **-load_logic full_interface** option setting.

-optimize_block_interface true | false

Specifies whether the interface logic for the block is optimized during top-level optimization.

By default, this option is set to **false** and the tool optimizes only the top-level logic, and not the interface logic. This option is only supported in Design Compiler in topographical mode for block abstractions that were created by Design Compiler in topographical mode.

-optimize_shared_logic true | false

Specifies whether the shared logic is optimized during interface optimization. The shared logic is defined as the list of cells and nets that are part of the interface timing paths for the block and are also shared with some block-level timing paths that are not visible at the top level.

Enabling this option can potentially disturb block-level register-to-register timing.

This option is applicable only when used with the **-optimize_block_interface** option.

By default, this option is **false** and interface optimization does not optimize the shared logic. This option is only supported in Design Compiler in topographical mode for block abstractions that were created by Design Compiler in topographical mode.

-size_only_mode off | auto | in_place | footprint

This option is currently ignored in Design Compiler because cell sizing is the only optimization that is performed.

-block_update_setup_script file_name

Specifies the setup file to be sourced during the block update of the specified block.

This file is useful for passing any variable settings for the block update phase. The file name should not include relative paths.

-block_update_cmd_script file_name

Specifies the script file with the commands to be run to update the specified block.

If this option is not specified, the tool automatically runs the **extract_rc -estimate** command for the block update.

If this option is specified, the tool sources the specified script file and does not run the **extract_rc -estimate** command. However, the tool still automatically runs **create_block_abstraction** and **write_file** after sourcing the script file.

The file name should not include relative paths.

-reset

Removes the top-level implementation options for the specified blocks.

DESCRIPTION

The **set_top_implementation_options** command sets the top-level implementation options for the specified blocks. Prior to this command, the blocks for which top-level implementation options are specified need to have block abstraction information annotated on them by using the **create_block_abstraction** command.

You must set the top-level implementation options for a block before loading it.

To allow top-level optimization to optimize the interface logic of the blocks, you must ensure the following:

- You are running Design Compiler in topographical mode.
- The block is not multiply instantiated in the design.
- The block abstraction was created using Design Compiler in topographical mode.

The option settings are not stored in the database. You need to specify these settings in every session.

To verify the top-level implementation options that are specified, see the report generated by the **report_top_implementation_options** command.

The following example enables block interface optimization for the blk1 block.

```
prompt> set_top_implementation_options -block_references blk1 \  
-optimize_block_interface true
```

The following example resets the top-level implementation options for the blk1 and blk2 blocks.

```
prompt> set_top_implementation_options \  
-block_references {blk1 blk2} -reset
```

SEE ALSO

check_block_abstraction(2)
create_block_abstraction(2)
report_block_abstraction(2)
report_top_implementation_options(2)

set_transform_for_retiming

Sets the **transform_for_retiming** attribute on cells in the current design. This can effect both hierarchical cells and sequential leaf cells.

SYNTAX

```
integer set_transform_for_retiming  
  cell_list  
  multiclass | decompose | dont_retime
```

Data Types

cell_list list

ARGUMENTS

cell_list

Specifies a list of cells on which the **transform_for_retiming** attribute is to be set. If you specify more than one cell name, the names must be enclosed either in quotation marks or in braces ({}).

multiclass | **decompose** | **dont_retime**

Specifies the value with which to set the **transform_for_retiming** attribute. There is no default value. One of the values must be specified.

- **multiclass** specifies that the cell will be moved together with any synchronous or asynchronous preset or clear or synchronous load-enable signals.
- **decompose** specifies that synchronous load-enable and synchronous reset logic are made explicit, and only the basic storage register is moved. For example, if a register has a data (D) input, a clock (CLK) input, and a synchronous clear (SD) input with active low polarity and the **transform_for_retiming** attribute is set to *decompose*, it is decomposed into a simple register with D and CLK input and an and gate driving the D input. The inputs of this and gate are the original data net and the synchronous clear net. Only the simple register will be moved for retiming, while the and gate stays in place.
- **dont_retime** specifies that the cell will not be retimed. It may still be mapped to a different library cell during incremental mapping optimizations.

DESCRIPTION

This command sets the **transform_for_retiming** attribute on cells in the current design.

If placed on a sequential, non-hierarchical cell, the attribute determines the way in which this cell will be transformed for retiming.

If you want to disable both retiming and sequential mapping optimizations for a cell, use the **dont_touch** attribute.

If the attribute is set on a hierarchical cell, it applies to all sequential cells in the hierarchy below this cell, unless the attribute is set on a hierarchical cell in between or on the sequential leaf cell itself. Values of the attribute set on lower levels of hierarchy override those set on higher levels.

The attribute does not effect non-sequential non-hierarchical cells.

If the attribute has neither been set on a sequential leaf cell nor on any of its hierarchical parent cells, the transform that is applied to this cell is determined by the corresponding command line option chosen for the **optimize_registers** command, or by the default setting for the retiming command that you are using.

If the **dont_touch** attribute is set to *true* on a cell, the **transform_for_retiming** attribute does not come into effect on this cell or any of its child cells.

Using the attribute can improve timing results at the cost of increased area for certain designs.

To remove the **transform_for_retiming** attribute, use **remove_attribute**.

EXAMPLES

The data inputs of two synchronous clear register cells R_1 and R_2 are connected to the same net, while the synchronous clear inputs are connected to two different and unrelated clear signals. If **optimize_registers** is invoked with the command line option -**sync_trans multiclass**, it will not move any of these registers backward over the driving cell of the net connected to the data ports, even if it would improve the timing of the design. Setting the **transform_for_retiming** attribute to *decompose* on the register cells allows backward movement to achieve better timing results. However, there are additional gates in the design. Use the following command to achieve this:

```
prompt> set_transform_for_retiming find(cell, {R_1, R_2}) decompose
```

The following command removes the attribute from the two cells:

```
prompt> remove_attribute find(cell, {R_1, R_2}) transform_for_retiming
```

SEE ALSO

balance_registers(2)
optimize_registers(2)
set_optimize_registers(2)
set_dont_touch(2)

set_unconnected

Lists output ports to be unconnected.

SYNTAX

```
int set_unconnected  
    port_list
```

Data Types

port_list list

ARGUMENTS

port_list

A list of port names in the current design that are to be left unconnected. If more than one port is specified, they must be enclosed either in quotes or in braces ({}).

DESCRIPTION

Assigns an **output_not_used** attribute to ports in *port_list*. This information is used by **compile** to create smaller designs by eliminating logic that drives an unconnected output port and therefore might not need to be maintained during optimization. After a design with an unconnected output port is compiled, the port usually is not driven by anything inside the design.

This attribute can be removed using the **remove_attribute** command.

Note:

The **set_unconnected** command cannot be used on input ports. Input ports can be tied to logic one, logic zero, or don't-care using **set_logic_one**, **set_logic_zero**, and **set_logic_dc**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies that output ports "CARRY_OUT" and "err_1" are not to be used.

```
prompt> set_unconnected {CARRY_OUT err_1}
```

SEE ALSO

- `get_attribute(2)`
- `remove_attribute(2)`
- `report_port(2)`
- `reset_design(2)`
- `set_logic_one(2)`
- `set_logic_zero(2)`
- `set_logic_dc(2)`
- `simplify_constants(2)`

set_ungroup

Sets the **ungroup** attribute on specified designs, cells, or references, indicating that they are to be ungrouped during **compile**.

SYNTAX

```
int set_ungroup  
    object_list true | false
```

Data Types

object_list list

ARGUMENTS

object_list

Specifies a list of cells, references, or designs to be ungrouped during **compile**.

true | false

The value with which to set the **ungroup** attribute. The default is **true**.

DESCRIPTION

Sets the **ungroup** attribute on the specified objects so that they are ungrouped (collapsed to one design level) before they are optimized. The **compile** command does not ungroup cells or designs with the **dont_touch** attribute.

This attribute is removed with the **remove_attribute** or **reset_design** commands.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, the **ungroup** attribute is set on the cell U1.

```
prompt> set_ungroup U1
```

If the **ungroup** attribute is specified on a reference object, cells using that reference are ungrouped during **compile**. In the following example, all the instances of ADDER in the current design are ungrouped.

```
prompt> set_ungroup [get_references ADDER]
```

A design with this attribute is ungrouped whenever **compile** encounters it.

```
prompt> set_ungroup [get_designs ND17]
```

Specifying **set_ungroup false** on a MUX operator sets a map_only and size_only on the MUX to ensure that the MUX is preserved.

SEE ALSO

- compile(2)
- current_design(2)
- remove_attribute(2)
- reset_design(2)
- ungroup(2)
- ungroup_keep_original_design(3)

set_uninitialized_register_value

Specifies the initial state to be set on uninitialized registers by constant removal engines. Uninitialized registers are registers for which the constant engines could not compute an initial state.

SYNTAX

```
int set_uninitialized_register_value  
  [-value init_value]  
  object_list
```

Data Types

object_list list

ARGUMENTS

-value *init_value*

Specifies the initial state value to be used by the constant engines (0 or 1).

object_list

Specifies the set of registers on which the value will be set as an initial state.

DESCRIPTION

Specifies a constant value (0 or 1) as the initial state for uninitialized registers listed in *object_list*.

EXAMPLES

The following example shows how to specify an initial state on the sequential cell U0.

```
prompt> set_uninitialized_register_value -value 0 U0  
prompt> set_uninitialized_register_value -value 1 U0
```

SEE ALSO

compile(2)
find(2)
get_attribute(2)
remove_attribute(2)

set_units

Sets the units used for resistance, capacitance, timing, leakage power, current, and voltage. The units must be consistent with the main library units.

SYNTAX

```
string set_units  
[-resistance ohm|kohm|mohm|Mohm|10ohm|100ohm]  
[-capacitance f|ff|pf|nf|uf|mf|10ff|100ff]  
[-time s|fs|ps|ns|us|ms|10ps|100ps]  
[-power w|fw|pw|nw|uw|mw|10uw|100uw|10mw|100mw|10pw|100pw|10nw|100nw]  
[-current A|fA|pA|nA|uA|mA|10uA|100uA|10mA|100mA]  
[-voltage v|fv|pv|nv|uv|mv|10mv|100mv]
```

ARGUMENTS

-resistance ohm|kohm|mohm|Mohm|10ohm|100ohm

Sets the resistance unit.

-capacitance f|ff|pf|nf|uf|mf|10ff|100ff

Sets the capacitance unit.

-time s|fs|ps|ns|us|ms|10ps|100ps

Sets the time unit.

-power w|fw|pw|nw|uw|mw|10uw|100uw|10mw|100mw|10pw|100pw|10nw|100nw

Sets the power unit.

-current A|fA|pA|nA|uA|mA|10uA|100uA|10mA|100mA

Sets the current unit.

-voltage v|fv|pv|nv|uv|mv|10mv|100mv

Sets the voltage unit.

DESCRIPTION

The **set_units** command sets the units for SDC file. The specified units cannot conflict with the main library units; therefore, you

should not use the **set_units** command interactively.

When the tool writes an SDC file, the **set_units** command is always written out as the first SDC command. If no units are set, the tool gets the units from the main library (the first link library).

EXAMPLES

The following example specifies the set_units command that is written to the SDC file.

```
prompt> set_units -time ns -resistance kohm -capacitance pf \  
-power uW -voltage V -current A
```

SEE ALSO

report_units(2)

set_unloaded_register_removal

Sets an attribute to unloaded register(s) to selectively preserve or optimize it.

SYNTAX

status **set_unloaded_register_removal**
objects *true | false*
[*flag*]

ARGUMENTS

DESCRIPTION

The **set_unloaded_register_removal** command is applied on unloaded sequential cell(s) or hierarchical cell(s) to optimize or preserve it selectively. It set an attribute **remove_unloaded_register** on specified register. The attribute can store true or false based upon your input. **set_unloaded_register_removal** when set to false on objects it sets the attribute **remove_unloaded_register** to value false and does not allow removal of the unloaded register. When the command is set to true on objects it allows optimization of those unloaded registers. You can use the attribute **remove_unloaded_register** to query the unloaded registers which are not removed after compile_ultra. This attribute can be used before compile_ultra to query registers which are to be removed. **set_unloaded_register_removal** honors size_only and issues warning if any register in this category is been set.

EXAMPLES

The following command optimizes register unld_regA:

```
prompt> set_unloaded_register_removal unld_regA true
```

The following command preserves register unld_regB:

```
prompt> set_unloaded_register_removal unld_regB false
```

The following command issues warning OPT-1222 as register unld_sizeonly_regC is size_only:

```
prompt> set_unloaded_register_removal unld_sizeonly_regC false
```

SEE ALSO

set_upf_cell_mismatch

Used to specify relaxations that tool can apply to get a final mapped netlist.

SYNTAX

```
status set_upf_cell_mismatch  
-allow_tls_violation  
-allow_pvt_mismatch  
-no_unmapped retention | isolation | both  
-reset
```

ARGUMENTS

-allow_tls_violation

When this option is specified, target library subset constraint may be violated for power management cells (retention, isolation and level shifter). If the tool is not able to map to a cell honoring target library subset, it will attempt to map to a cell violating target library subset constraint.

-allow_pvt_mismatch

When this option is specified, PVT matching may be relaxed for power management cells (retention, isolation and level shifter). Design Compiler will not complain if there is PVT mismatch during linking and mapper may bring in PVT mismatching cells in case matching cells are not available. PVT matching relaxation will eventually relax process, voltage, temperature, rail voltages, target_library_subset and bias requirements. As target_library_subset is also relaxed here, -allow_pvt_mismatch is a superset of -allow_tls_violation.

-no_unmapped retention | isolation | both

When this option is specified, UPF constraint may be relaxed such that the final netlist will not have any unmapped PM cells.

If the option value is specified as **retention**, then in the absence of a suitable Retention library cell, tool will relax the retention constraint on unmapped sequential element and it will be mapped to non-Retention cell.

If the option value is specified as **isolation**, then in the absence of a suitable Isolation library cell, tool will relax the Isolation constraints specified in the UPF and unmapped isolation cells will be removed from the final netlist.

If the option value is specified as **both**, unmapped isolation cells will be removed and unmapped sequential cells with retention constraints will be mapped to non-retention.

-reset

When this option is specified, all the previously specified relaxations (-allow_tls_violation, -allow_pvt_mismatch and -no_unmapped) will be reset. The benefit of this option is that once user brings in all the required libraries, the existing relaxations can be reset using this option.

DESCRIPTION

This command allows the user to specify relaxation on `target_library_subset`, PVT matching and having a netlist free of unmapped PM cells.

This is a cumulative command and user can have multiple specifications.

All the options are optional. If just `set_upf_cell_mismatch` is specified without any options, then it means all the relaxations are enabled.

This command always applies to the entire current design.

The netlist synthesized with these relaxations might have QoR degradation and verification failures.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to specify relaxations using `set_upf_cell_mismatch` command.

```
prompt> set_upf_cell_mismatch -allow_tls_violation  
prompt> set_upf_cell_mismatch -allow_pvt_mismatch
```

This command will be interpreted as

```
set_upf_cell_mismatch -allow_tls_violation -allow_pvt_mismatch
```

SEE ALSO

set_upf_query_options

Sets the options for specifying design elements in UPF commands.

SYNTAX

status **set_upf_query_options**
-bus_struct_mode true | false

ARGUMENTS

-bus_struct_mode true | false

Enables bus/struct querying support for the **-elements** option of the **set_retention**, **set_isolation**, and **map_retention_cell** commands. The default behavior is **false**.

DESCRIPTION

The **set_upf_query_options** command sets the mode for specifying bus/struct elements.

When the **-bus_struct_mode** argument is set to **true**, it allows vector elements and struct elements to be referenced by their RTL vector/struct names in the **-elements** option of the **set_retention**, **set_isolation**, and **map_retention_cell** commands.

For example, given an 8-bit register named ROUT:

```
reg [7:0] ROUT;
```

The **set_retention** command can refer to ROUT as follows:

```
prompt> set_retention -elements {ROUT} ...
```

This avoids the use of wildcard characters, which require more runtime resources to process:

```
prompt> set_retention -elements {ROUT*} ...
```

while avoiding the need to enumerate all bits of the register:

```
prompt> set_retention -elements {ROUT_reg[0] ROUT_reg[1] ... ROUT_reg[7]} ...
```

To use this option, the following application variables must be appropriately set prior to reading in RTL:

```
prompt> set_app_var bus_naming_style = "%s[%d]" (default setting)
prompt> set_app_var hdlin_enable_upf_compatible_naming true
```

Furthermore, this option should only be set to **true** while loading the initial RTL UPF, and set to **false** at other times, as demonstrated in the EXAMPLES section.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows an instance of a user-defined struct, "ip_n", and a 8-bit bused register "rout" that is used in the **-elements** option of **set_retention** command.

```
typedef struct packed {
    logic [7:0] pad;
    logic [7:0] offset;
    logic B;
} user_struct;

module top(..., rout);
    output logic [7:0] rout;
    user_struct IP_N;
    ...
}

prompt> set_hdlin_enable_upf_compatible_naming true
prompt> read_sverilog top.sv
prompt> get_cells rout*
{rout_reg[7] rout_reg[6] rout_reg[5] rout_reg[4] rout_reg[3] rout_reg[2] rout_reg[1] rout_reg[0]}
prompt> get_cells ip_n_reg.*
{ip_n_reg.pad[7] ... ip_n_reg.pad[0] ip_n_reg.offset[7] ... ip_n_reg.offset[0] ip_n_reg.B}
prompt> set_upf_query_options -bus_struct_mode true
...
prompt> set_retention -domain TOP -retention_power_net VDD \
    -retention_ground_net VSS -elements {rout} RETN_OUT
prompt> set_retention -domain TOP -retention_power_net VDD \
    -retention_ground_net VSS -elements {ip_n.pad} RETN_IP_N_PAD
...
prompt> set_upf_query_options -bus_struct_mode false
...
```

SEE ALSO

load_upf(2)
 set_retention(2)
 set_isolation(2)
 map_retention_cell(2)
 hdlin_enable_upf_compatible_naming(3)

set_user_attribute

Sets the value of an attribute on a design or library object.

SYNTAX

```
list set_user_attribute  
  object_list  
  attribute_name  
  attribute_value  
  [-bus]  
  [-quiet]
```

Data Types

<i>object_list</i>	list
<i>attribute_name</i>	string
<i>attribute_value</i>	attribute value

ARGUMENTS

object_list

Specifies a list of design or library objects on which the attribute is to be set. For details on searching for design and library names, see the man pages for the **get_designs** and **get_libs** commands.

attribute_name

Specifies the name of the attribute to set.

attribute_value

Specifies the value of the attribute. The data type must be the same as the attribute data type.

-bus

Sets the attribute on the bus as a whole, and not on each individual bus member.

-quiet

Turns off warning messages that would otherwise be issued if the attribute or objects are not found.

DESCRIPTION

The **set_user_attribute** command sets the value of an attribute on an object. It is the same as the **set_attribute** command. See the **set_attribute** command man page for further information.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

remove_user_attribute(2)
set_attribute(2)
attributes(3)

set_user_budget

Sets user budgets or budget ratios.

SYNTAX

```
string set_user_budget  
-from object_list  
-to object_list  
[-percent]  
value
```

Data Types

<i>object_list</i>	list
<i>value</i>	float

ARGUMENTS

-from *object_list*

Indicates the pins or clocks from which the budget is specified.

-to *object_list*

Indicates the pins or clocks to which the budget is specified.

-percent

Indicates the specified budget value is a percentage.

value

Sets the budget value or percentage.

DESCRIPTION

The **set_user_budget** command is used to specify user directives for budget allocation. You can specify budgets as an absolute value or as a percentage of available constraint to be allocated. These directives are honored by budget allocation. If you don't specify user budgets for a budgeted cell, budget allocation is performed automatically.

The objects specified in either the **-from** or **-to** options must be inputs pins or output pins of budgeted cells. User budgets specified for cells which are not ultimately budgeted are ignored.

EXAMPLES

The following command specifies the budget from clock clk1 to the two outputs is 5.0.

```
prompt> set_user_budget -from clk1 -to {l2/out1 l2/out2} 5.0
```

The following command specifies that the budget from the IN input to the Y of l1 should be 70 percent of the path constraint.

```
prompt> set_user_budget -from l1/IN -to l1/Y -percent 70.0
```

SEE ALSO

`dc_allocate_budgets(2)`

`report_path_budget(2)`

set_utilization

Specifies placement utilization constraint for core area. This command is supported only in topographical mode.

SYNTAX

```
int set_utilization
    util_float
```

Data Types

util_float float

ARGUMENTS

util_float

Specifies target placement utilization for core area. The utilization is defined as standard cell utilization:

$$\text{total\_standard\_cell\_area} / (\text{core area} - \text{blockage area} - \text{macro\_cell\_area})$$

Area Definition ----

Core area Core area bounding box

Standard cell sum of fixed and non-fixed area standard cell area

blockage area sum of placement keepout area and filler cell (physical only cell) area

macro_cell area sum of fixed and non-fixed macro cell area

Note: for utilization calculation purpose, we define the macro cell as the cell in the netlist without lib site linked, or the cell width and height is larger than 5 minimum lib site height.

Value for *util_float* is a float number less than 1, for example, 0.85 for 85% utilization. The default value is 0.60.

DESCRIPTION

This command defines design level physical constraint for utilization. The constraint will be used to generate coarse floorplan during synthesis.

Use this command after loading the physical library and design, but before running synthesis.

There are also other commands that set constraints on core area, such as **set_placement_area**, **set_rectilinear_outline** and

set_aspect_ratio. Among the four commands, **set_placement_area** has the highest priority, while **set_rectilinear_outline** has higher priority than **set_utilization** and **set_aspect_ratio**. This means if the placement area constraint is set by **set_placement_area** or rectilinear outline constraint is set by **set_rectilinear_outline** on the current design, then the utilization constraint will not be used during synthesis. Furthermore, in this case, utilization will be neither reported by the command **report_physical_constraints**, nor written out by the command **write_physical_constraints**.

EXAMPLES

The following example shows how to set the floorplan utilization to 80%.

```
prompt> set_utilization 0.8
```

SEE ALSO

`set_placement_area(2)`
`report_physical_constraints(2)`
`write_physical_constraints(2)`

set_verification_priority

Sets the **verification_priority** attribute on specified designs, cells, or references, indicating the level of optimization during **compile_ultra**.

SYNTAX

```
status set_verification_priority  
-all  
[-high]  
object_list
```

Data Types

object_list list

ARGUMENTS

-all

Sets the attribute on all designs (equivalent to **[get_designs *]**).

-high

Sets the verification priority level. This option maps to the low effort of the optimization effort level.

object_list

Specifies a list of cells, references, or designs on which the attribute is to be set.

DESCRIPTION

This command sets the **verification_priority** attribute on the specified objects. During **compile_ultra**, the optimizations of the design with the attribute are adjusted depending on the verification priority level so that the potential for hard verification is reduced. It primarily adjusts the optimizations on datapath logic. Ungrouping of DesignWare components in the design with the attribute is disabled. It also identifies CRC logic in the design and isolates it by creating the new hierarchy. The attribute is not transitive to its subdesigns. Design Compiler does not ungroup cells or designs on which the **verification_priority** attribute is set.

When the attribute is set on a synthetic operator, such as an adder or a multiplier, the resource sharing and the datapath extraction is not performed on the operator. The DesignWare hierarchy of the operator is preserved, except that the size of the DesignWare is small.

You can use the **remove_verification_priority** command to remove the **verification_priority** attribute.

EXAMPLES

In the following example, the **verification_priority** attribute is set to **medium** on the *U1* cell:

```
prompt> set_verification_priority U1
```

If the **verification_priority** attribute is specified on a reference object, cells using that reference are not ungrouped. In this example, all instances of **ADDER** in the current design are not ungrouped:

```
prompt> set_verification_priority [get_references ADDER]
```

SEE ALSO

- `compile(2)`
- `current_design(2)`
- `hdlin_verification_priority(3)`
- `remove_verification_priority(2)`

set_verification_top

SYNTAX

status **set_verification_top**

ARGUMENTS

None

DESCRIPTION

This command is a part of the recommended Synopsys verification setup (SVF) creation flow.

This recommended flow is also called the Guide Hierarchical Map (GHM) flow because it improves how complex hierarchy transformations are handled in the verification guidance.

For more information on this flow, see the **hdlin_enable_hier_map** man page.

This command requires that the **hdlin_enable_hier_map** application variable was set to **true** prior to reading any RTL files.

If you attempt some post-design-read command that would modify the design, but you have forgotten to run the **set_verification_top** command, the tool issues a GHM-003 error. Any subsequent attempts to run the **compile** or **compile_ultra** command will also fail.

EXAMPLES

The following script shows the correct command ordering for this recommended flow.

```
# Specify SVF output file
prompt> set_svf ./guidance/top.svf

# Enable GHM flow
prompt> set_app_var hdlin_enable_hier_map true

# Read design files
prompt> analyze -format sverilog {sub.sv top.sv}
prompt> elaborate top
```

prompt> **current_design top**

Issue this command after reading last RTL file and before
other commands that can modify netlist

prompt> **set_verification_top**

SEE ALSO

hdlin_enable_hier_map(3)

hdlin_ignore_ghm_errors(3)

set_via_ladder_candidate

Specify via ladder candidates for optimization.

SYNTAX

```
status set_via_ladder_candidate  
  library_pin  
  -ladder_name ladder_name
```

Data Types

```
library_pin  collection  
ladder_name string
```

ARGUMENTS

library_pin

Specifies library pin on which to set via ladder candidates.

This is a required option.

-ladder_name *ladder_name*

Specifies the name of the via ladder to set on library pin.

This is a required option.

DESCRIPTION

This command sets the list of via ladder candidates on library pins for via ladder optimization.

If the variable *spg_enable_via_ladder_opto* is turned on for via ladder optimization, the compile will infer via ladders on pins to improve the timing based on the via ladder candidates on their library pins.

The priority of via ladder candidates follows their order on library pin, the first via ladder with timing improvement has higher priority.

EXAMPLES

The following command shows setting typical via ladders on pins.

```
prompt> set_via_ladder_candidate [get_lib_pin slow/CLKBUFX20/Y] -ladder_name VL1_2 -cell_spacing {200.000 220.000}
```

SEE ALSO

report_via_ladder_candidates(2)
spg_enable_via_ladder_opto(3)

set_via_ladder_constraints

Attaches ordered via ladder lists to specified instances of logical pins.

SYNTAX

```
status set_via_ladder_constraints  
-pins pin_names  
  via_ladder_list
```

Data Types

<i>pin_names</i>	collection
<i>via_ladder_list</i>	list

ARGUMENTS

-pins *pin_names*

Specifies the name of the logical pins to set via ladder constraints.

This is a required option.

via_ladder_list

Specifies the ordered via ladder list to be attached to the specified logical pins.

A via ladder is a physical structure of connected vias and wires that is placed on top of a standard cell pin as a special connecting device. A via ladder can have, for example, two rows of via1, four columns of via2, and one row of via3 connected with a metal4 wire.

A named via ladder is defined by set of rules in the technology file.

This is a required argument.

DESCRIPTION

This command sets the list of via ladder names as via ladder constraints for specified pins. During RC extraction the first appropriate via ladder from the list will be used for RC estimation on the specified pin.

EXAMPLES

The following command shows setting typical via ladders on pins

```
prompt> set_via_ladder_constraints -pins [get_pins */Z] {VL3_3 VL3_2 ABC DE}
```

SEE ALSO

[remove_via_ladder_constraints\(2\)](#)

[report_via_ladder_constraints\(2\)](#)

set_via_ladder_rules

Sets via ladder rules for the design.

SYNTAX

```
status set_via_ladder_rules
-master_pin_map masterpin_ladders_list
-master_pin_map_file file
-default_ladders via_ladders_list
-all_instances_of cell_master_pin_list
-all_clock_outputs true | false
-all_clock_inputs true | false
-all_pins_driving port_list
-remove_all_rules
```

Data Types

```
masterpin_ladders_list  string
file                    string
cell_master_pin_list    string
port_list               port names or collection
```

ARGUMENTS

-master_pin_map masterpin_ladders_list

Specifies what type of ladder(s) should be used on an instance of given cell_master/pin if it is to get a ladder. Format of masterpin_ladders_list is {{cell_master/pin {via_ladder_list}}...}. The cell_master should be a library cell, entries where cell_master is not a library cell would be ignored with a warning.

-master_pin_map_file file

Specifies file name to provide entries for -master_pin_map option. Format of file entry is same as needed in option -master_pin_map. master_pin_map entries can be provided with both the options together. In case of clash, entry in file will be given priority. The cell_master should be a library cell, entries where cell_master is not a library cell would be ignored with a warning.

Design Compiler topographical mode does not support this option.

-default_ladders via_ladders_list

Specifies what ladder to use if a cell/pin requires a ladder based on a rule, but it is not defined in the master_pin_map list. Format of via_ladders_list is {VL1 VL2 ...}.

Design Compiler topographical mode does not support this option.

-all_instances_of_cell_master_pin_list

Specifies that anytime an instance of this cell master/pin is used, a ladder is to be created automatically. Format of cell_master_pin_list is {cell_master1/pin1 cell_master2/pin2 ...}. The cell_master should be a library cell, entries where cell_master is not a library cell would be ignored with a warning.

Design Compiler topographical mode does not support this option.

-all_clock_outputs true | false

Specifies that all output pins on clock nets get a ladder. Default values is false.

Design Compiler topographical mode does not support this option.

-all_clock_inputs true | false

Specifies that all input pins on clock nets get a ladder. Default values is false.

Design Compiler topographical mode does not support this option.

-all_pins_driving port_list

Specifies that all pins driving one of the ports in the collection gets a ladder.

Design Compiler topographical mode does not support this option.

-remove_all_rules

If this option is specified, all the rules will be removed.

DESCRIPTION

This command sets via ladder rules for the block. These rules will be used by zroute to automatically update via ladders as the design changes. Instance/pin specifications set with via ladders constraints commands will remain, and can/should still be used for fixed/non-changing cells. These rules are more valuable for dynamically changing cells. If both an instance specific constraint and a rule exist, the instance constraint should apply.

This command returns 1 if succeeded, 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information if there is no scenario dependency for the options.

EXAMPLES

The following examples associate via ladder 'VP1' with library pin 'BUFFD10/Z' and 'VP2' with 'INVD8/ZN'

```
prompt> set_via_ladder_rules -master_pin_map { {lib/BUFFD10/Z {VP1}} {lib/INVD8/ZN {VP2}} }
```

```
1
```

SEE ALSO

- `remove_via_ladder_rules(2)`
- `report_via_ladder_rules(2)`
- `set_via_ladder_constraints(2)`
- `remove_via_ladder_constraints(2)`
- `report_via_ladder_constraints(2)`

set_voltage

Applies an operating voltage to a list of supply net or internal supply port objects.

SYNTAX

```
status set_voltage  
    max_voltage  
    [-min min_voltage]  
    -object_list supply_objects
```

Data Types

<i>max_voltage</i>	float
<i>min_voltage</i>	float
<i>supply_objects</i>	collection

ARGUMENTS

max_voltage

Specifies the operating voltage for the maximum (worst) case.

-min *min_voltage*

Specifies the operating voltage for the minimum (best) case.

-object_list *supply_objects*

Specifies the supply nets and internal supply ports that have the operating voltages specified in this command.

DESCRIPTION

Defines the operating voltage on the supply nets and their corresponding power nets so that the parts of the design powered by these supply nets or power nets are timed and optimized at the specified voltage.

It is an error if supply nets do not have any voltage set on them.

To see the operating voltages of the supply nets, use the **report_supply_net** command.

The objects can also be supply ports that correspond to the internal PG pins of low-power macros with fine-grain switch cells. The internal PG pins must satisfy the following conditions:

- It belongs to a macro cell whose **switch_cell_type** attribute has a value of "fine_grain".
- The pin's **direction** attribute has a value of "internal".
- The pin's **pg_type** attribute has a value of "internal_power" or "internal_ground".
- The pin has a **switch_function** attribute.
- The pin has a **pg_function** attribute.

If there is no voltage set on the supply port, the voltage value is picked from the voltage name of its corresponding internal PG pin, as defined in the Liberty file of the macro.

A voltage can be set on the supply port if its internal PG pin derives a different voltage than the primary voltage for the macro. This voltage is then used for linking and timing calculation purposes.

To see the operating voltages of the supply ports, use the **report_supply_port** command.

The operating voltage of a supply object, once defined, can only be overridden with another **set_voltage** command.

If voltage ranges are specified for a power net, the operating voltages of the power net must fall within one of the specified ranges. Furthermore, the minimum voltage must fall within the same range as the maximum voltage.

The operating voltage of a power net, once defined, can only be overridden with another **set_voltage** command. It can also be cleared by using the **reset_design** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example shows a typical usage of the **set_voltage** command.

This example creates a power domain named CNTL that has two supply nets, sn1 and sn2. Both supply nets are assigned a worst case voltage of 0.9. A worst case voltage of 0.9 is also set on the supply port corresponding to the internal VDDPI PG pin of the ram macro cell, which has a fine-grain switch. The definition for the VDDPI pin in the Liberty file is

```
pg_pin(VDDPI) {
  voltage_name : VDD;
  direction : internal;
  pg_type : internal_power;
  switch_function : "CTRL";
  pg_function : "VDDP";
}
```

where VDD is a voltage value defined in the voltage map; CTRL is a switch-pin that controls the fine-grain switch of the macro; and VDDP is a PG pin of type primary_power.

The following commands perform these assignments:

```
prompt> create_power_domain CNTL
1
prompt> create_supply_net sn1 -domain CNTL
1
prompt> create_supply_net sn2 -domain CNTL
```

```
1
prompt> set_voltage 0.9 -object_list {sn1 sn2}
1
prompt> set_voltage 0.9 -object_list {ram/VDDPI}
1
```

SEE ALSO

- create_supply_net(2)
- create_power_domain(2)
- set_domain_supply_net(2)
- connect_supply_net(2)
- report_supply_net(2)
- report_power_domain(2)

set_voltage_model

Sets the voltage model for the specified library.

SYNTAX

```
status set_voltage_model
      library_name
      [-voltage {voltage_name voltage_value}]
```

Data Types

<i>library_name</i>	string
<i>voltage_name</i>	string
<i>voltage_value</i>	float

ARGUMENTS

library_name

Specifies the name of the library for which to define the voltage model.

-voltage { *voltage_name* *voltage_value* }

Sets the voltage name and value pair for the specified library. You can specify this option multiple times in a single run of the command.

DESCRIPTION

The **set_voltage_model** command updates the voltage map of the library. If the voltage is already defined in the library, the new value overrides the old value. If the voltage is not defined, the new definition is added to the library voltage map.

EXAMPLES

The following example defines the voltage model for a single-rail library.

```
prompt> set_voltage_model standard_lib \
-voltage {VDD 1.2} \
-voltage {VSS 0.0}
```

The following example defines the voltage model for a multiple-rail library.

```
prompt> set_voltage_model standard_lib \  
  -voltage {VDD 1.2} \  
  -voltage {VSS 0.0} \  
  -voltage {VDD_BACK 0.7}
```

SEE ALSO

`write_lib_specification_model(2)`
`update_lib_model(2)`

set_vsdc

Generates a setup information file in V-SDC format for efficient compare point matching in formal verification tools.

SYNTAX

Boolean **set_vsdc**
filename
[-append]
[-off]

ARGUMENTS

filename

Names the file into which verification setup information will be recorded. You must specify a file name unless the *-off* option is specified.

-append

Appends to the specified file. If another verification setup file is already open then it will be closed before opening the specified file. If *-append* is not used then **set_vsdc** will overwrite the named file, if it exists.

-off

Stops recording verification setup information to the currently-open file. To resume recording into the same file, you must re-issue the **set_vsdc** command with the *-append* option.

DESCRIPTION

This command causes Design Compiler to start recording setup information for formal verification tools. (If you are using Formality, the Synopsys formal verification product, you should use the *set_svf* command, instead.) The V-SDC file that is produced may be used by some verification tools during the matching step to facilitate the alignment of compare points. By using the automatically-generated V-SDC file, the user is relieved of the time-consuming and error-prone process of entering the setup information manually.

The V-SDC file is a series of Tcl commands stored in plain-text format.

The V-SDC file is used to account for name changes that occur during synthesis as a result of running commands such as *group*, *ungroup*, *uniquify* and *change_names*. Also, certain operations performed by the *compile* command perform operations that are recorded in the V-SDC file.

Once this command is issued, Design Compiler will begin recording all relevant operations. Because information is buffered

internally, the file may not be complete until recording is stopped. Recording is stopped by running "set_vsdc -off" or by exiting dc_shell. Running **set_vsdc** with a new file name will write out and close the original V-SDC file before starting the new one.

This command returns a Boolean value indicating whether **set_vsdc** succeeded (true) or not (false).

EXAMPLES

The following example shows a typical invocation of the command

```
prompt> set_vsdc my_design.vsd
```

In the following example the verification setup information is appended to the existing V-SDC file, barfile.vsd.

```
prompt> set_vsdc -append barfile.vsd
```

SEE ALSO

- set_svf(2)
- ungroup(2)
- group(2)
- uniquify(2)
- change_names(2)
- compile(2)
- compile_ultra(2)
- fsm_auto_inferring(3)
- compile_seqmap_propagate_constants(3)

set_wire_load

Sets the wire loading model for the current design or for the specified cluster or ports.

SYNTAX

```
int set_wire_load  
  [-mode mode_name]  
  [-size value]  
  [-library library_name]  
  [-cluster cluster_name]  
  [-selection_group group_name]  
  [-min]  
  [-max]  
  [-name model_name]  
  [object_list]
```

Data Types

<i>mode_name</i>	string
<i>value</i>	float
<i>library_name</i>	string
<i>cluster_name</i>	string
<i>group_name</i>	string
<i>model_name</i>	string
<i>object_list</i>	string

ARGUMENTS

-mode *mode_name*

Specifies the mode to use to handle hierarchical wire load models; overrides the default mode, which is discussed in the DESCRIPTION section. Allowed values are *top*, *enclosed* or *segmented*.

-size *value*

Specifies the minimum block area of the design and all its subdesigns, to use when selecting the appropriate wire load. When you specify a *mode_name* of *enclosed*, the **-size** option can be supplied to control the automatic wire load selection based on block area for this design and for all of its subdesigns.

-library *library_name*

Specifies the name of the library that contains the desired wire load model.

-cluster *cluster_name*

Specifies the name of a cluster associated with the current design, to which this wire load model is to be assigned.

-selection_group *group_name*

Specifies the name of the selection group to use when determining the wire load model based on design or cluster area, when the **auto_wire_load_selection** variable is set to true. This option is required only when the library has more than one selection group defined and the default group defined in the library is not the desired group.

You can define a selection group for a design or for a cluster, but not for a port list. This option is supported only for the *enclosed* wire load mode, specified using the **-mode** option.

-min

Specifies which model to use for minimum delay analysis: use the wire load model or use the selection group. You cannot use the **-min** option to set a different wire load mode or minimum block size, because both minimum and maximum delay analysis always use the same values for these parameters.

-max

Specifies which model to use for maximum delay analysis: the wire load model or the selection group. Any model set for minimum delay analysis is not affected.

-name *model_name*

Specifies the wire load model to use. The model must be defined in the specified library or in one of the libraries in the **link_library** or the **local_link_library**).

object_list

Collection of objects that use the wire load model.

DESCRIPTION

This command has been obsoleted and is replaced by **set_wire_load_model**, **set_wire_load_mode**, **set_wire_load_selection_group**, and **set_wire_load_min_block_size**.

In its simplest usage, the **set_wire_load** command specifies a wire load model for the top-level design. This model is applied to all nets in that design, both top-level and those under the hierarchy.

You can use this command to set a port's external **wire_load_model**. The wire load model set on a port overrides the wire load model of the design for the net connected to that port.

You also can use this command to set a cluster's **wire_load_model**.

If no wire load model is specified for a design, a default is selected. Design Compiler searches the first library on the link path (or the first library in the design's **local_link_library**) for a default wire load to use. If the library has **wire_load_selection** tables, the cell area of the current design is used to automatically select the appropriate wire load model from the appropriate **wire_load_selection** table. If the library has more than one table, it typically specifies that one of the tables is the default (by setting the **default_wire_selection_group** attribute). You can override this value by using the **-selection_group** option. If you did not supply the **-min_block_size** option, the cell area of the design is assumed to be at least as large as the value specified for this option when selecting the appropriate wire_load to use. If there are no **wire_load_selection** tables, a **default_wire_load** is used if there is one in the library. If the library has neither of the aforementioned, no wire load model is used.

The **report_design** command displays the wire loading model and mode for the current design, and the libraries to which the current design is linked. The **report_clusters** command shows the wire load associated with any particular cluster, while the use of the **report_wire_load** command can provide a more detailed report on the **wire_load_models** associated with either a design or cluster. The **report_lib** command shows the wire loading models that are defined in the specified library, as well as any

wire_load_selection tables that might be defined there. The you can use **report_port** command to display the wire_load_model of the ports of a design.

To specify distinct wire load models for various hierarchical cells in the top-level design, specify wire load models on the designs referenced by these hierarchical cells, because wire load models are reference-specific and not instance-specific. You must also ensure that the wire_load_mode of the topmost design is not *top*; otherwise, the wire load models selected for lower-level blocks in the design are ignored. The same holds true when specifying a wire load model on a cluster; that is, the wire load mode of the top-level design must not be mode *top*, or these annotations are ignored.

If you don't specify a **wire_load_model_mode** for a design, a default **wire_load_model_mode** is searched for in the first library in the link path during **compile**. If that library does not have a default **wire_load_model_mode**, *top* is assumed. You can override a mode by using the **-mode** option. If the mode for the top-level design is *top*, the wire load model on subdesigns and subclusters has no affect, and the top-level wire load model is used to compute wire capacitance for all hierarchical or top-level nets within a top-level design. If the first library in the link path during **compile** has a wire_load_selection table, and if the mode is *top* and no wire load has been explicitly set for the topmost design, a wire load model is automatically selected based on the total cell area of the design, and is used for all designs in the hierarchy. For designs with an area smaller than the minimum specified, the **-min_block_size** option can set a minimum area to influence which wire load is selected.

If the mode for the top-level design is either enclosed or *segmented*, wire load models on subdesigns and subclusters are used to calculate wire capacitance in nets inside these blocks. If a wire load model is not specified for a subdesign, and if the first library in the link path contains a wire_load_selection table, the **compile** command automatically selects a wire load model to use based on the cell area of the subdesign. If you don't define a wire_load_selection table, the wire load model of the parent design or cluster is used. If the top-level mode is enclosed or segmented, the wire load model of the subdesign is not set by **characterize** unless the subdesign does not have a wire load.

If the model mode is enclosed, the wire load model of the smallest design that encloses a net completely is used to calculate the wire capacitance of that net. The number of pins on the net equals the number of leaf pins on the net.

If a model mode is segmented, a net is broken into segments; one segment within each hierarchical design. The wire capacitance of a net is the sum of wire capacitances on each segment. When calculating the wire capacitance of each segment, the hierarchical boundary pins are included in the total pin count for that segment. The segmented mode is not supported for wire_load_models on clusters.

Hierarchical wire load models are explained previously in terms of wire capacitance, but the wire resistance and wire area are also calculated according to these rules.

With the **-min** option, you can use **set_wire_load** to specify a different wire load model or wire load selection group to use for minimum delay analysis. Unless explicitly specified, minimum delay analysis uses the same wire loads that maximum delay analysis uses. Different wire loads for minimum delay analysis can be set on designs, ports, or clusters. It is not possible to specify a different wire load mode or a different minimum block size for minimum delay analysis only. In these cases these same value is always used for both minimum and maximum analysis.

Wire area is always calculated using the same wire loads used for maximum delay analysis.

Wire loading models contain all the information required by **compile** to estimate interconnect wiring delays. Following is an example of a wire loading model definition, as it appears in the source text of a library.

```
wire_load("90x90") {
  resistance : 0 ;
  capacitance : 1 ;
  area : 0 ;
  slope : 1.64 ;
  fanout_length( 1 , 1.9 ) ;
  fanout_length( 2 , 2.8 ) ;
  fanout_length( 3 , 4.7 ) ;
}
```

The name of this model is "90x90". The fields in this model are defined as follows:

resistance

The resistance per unit-length of interconnect wire.

capacitance

The capacitance per unit-length of interconnect wire.

area

The net-area per unit-length of interconnect wire.

fanout_length (fanout1, length1)

These pairs of numbers specify a lookup table that the design and dft compilers use to estimate the length of interconnect wire being driven by a cell with a given fanout. You must specify at least one of these pairs (with fanout = 1), and can specify as many additional pairs as necessary to characterize the desired fanout-length behavior. Linear interpolation between the two nearest pairs is used to estimate any points that are not in the lookup table.

slope

This value is used to characterize linear fanout or length behavior beyond the last pair specified in the lookup table.

To remove a wire load model from a design, use the **set_wire_load** command with no parameters, or use the **reset_design** command.

EXAMPLES

The following example sets the wire load model on the top-level design, on which there is no **local_link_library** set.

```
prompt> report_lib tech_lib
```

```
.
.
.
```

Wire Loading Model:

```
Name Library Res Cap Area Slope Fanout Length
-----
05x05 tech_lib 0.0000 1.0000 0.0000 0.1860 10 0.3900
10x10 tech_lib 0.0000 1.0000 0.0000 0.3110 1 0.5300
```

```
.
.
.
```

```
prompt> link_library = {tech_lib.db}
prompt> set_wire_load "10x10"
```

In the following example, the **-library** option specifies a wire load model of "10x10" from the library "my_lib.db".

```
prompt> set_wire_load "10x10" -library my_lib.db
```

The following example sets wire load model "10x10" on the subdesign "LOW" to be used for maximum and minimum delay analysis. Then, the wire load model is set to "20x20" and the wire load model mode is set to *enclosed* for the top-level design "TOP". Finally a different wire load model "20x20MIN" is set at the top level to be used for minimum delay analysis only.

```
prompt> current_design LOW
```

```
prompt> set_wire_load "10x10"
```

```
prompt> current_design TOP
```

```
prompt> set_wire_load "20x20" -mode enclosed
```

```
prompt> set_wire_load "20x20MIN" -min
```

The following example sets no wire load model on the subdesign "LOW", leaving it to be selected based on its cell area. Then, the wire load model is set to "20x20" and the wire load model mode is set to *enclosed* for the top-level design "TOP", with a **min_block_size** argument specifying to assume (in this case, LOW) a block size of at least 200.0 for all subdesigns.

```
prompt> current_design LOW /* no wire-load set now */
```

```
prompt> current_design TOP
```

```
prompt> set_wire_load "20x20" -mode enclosed -min_block_size 200.0
```

The following example sets wire load model "10x10" on the cluster 'CACHE'. Then, the wire load model is set to "20x20" and the wire load model mode is set to *enclosed* for the top-level design "TOP".

```
prompt> current_design TOP
```

```
prompt> set_wire_load -cluster CACHE "10x10"
```

```
prompt> set_wire_load "20x20" -mode enclosed
```

The following sequence of commands describes the external fanout of a port.

```
prompt> set_wire_load -port_list find(port, "O1") "default_wl"
```

```
prompt> set_load -fanout_number 5 find(port, "O1")
```

The following example sets the selection group for the current design.

```
prompt> current_design TOP
```

```
prompt> set_wire_load -selection_group 2layermetal
```

SEE ALSO

```
characterize(2)
current_design(2)
report_lib(2)
reset_design(2)
set_load(2)
set_local_link_library(2)
set_wire_load_model(2)
set_wire_load_mode(2)
set_wire_load_selection_group(2)
set_wire_load_min_block_size(2)
link_library(3)
```

set_wire_load_min_block_size

Sets the wire load **min_block_size** attribute on the current design.

SYNTAX

```
int set_wire_load_min_block_size  
    size
```

Data Types

size float

ARGUMENTS

size

A positive value in the units of the technology library, that specifies the minimum block area of the design and all its subdesigns, to use when selecting the appropriate wire load.

DESCRIPTION

You can use this command only if the **wire_load_model_mode** attribute has been set to *enclosed* using the **set_wire_load_mode** command.

Sets the wire load minimum block area for the current design and all its subdesigns. By default, the cell area of the design is assumed to be at least as large as the value specified for this option when selecting the appropriate wire load to use in **set_wire_load_model** command. For designs with an area smaller than the minimum specified, this command can set a minimum area to influence which wire load is selected. If there are no **wire_load_selection** tables, a **default_wire_load** is used if there is one in the library. If the library has neither of the aforementioned, no wire load model is used.

EXAMPLES

The following example sets no wire load model on the subdesign "LOW", leaving it to be selected based on its cell area. Then, the wire load model is set to "20x20" and the wire load model mode is set to *enclosed* for the top-level design "TOP", with a **min_block_size** argument specifying to assume a block size of at least 200.0 for all subdesigns (in this case, LOW).

```
prompt> current_design LOW /* no wire-load set now */
```



```
prompt> current_design TOP
```

```
prompt> set_wire_load_mode enclosed
```

```
prompt> set_wire_load_min_block_size 200.0
```

```
prompt> set_wire_load_model -name "20x20"
```

SEE ALSO

characterize(2)

current_design(2)

report_lib(2)

reset_design(2)

set_wire_load_model(2)

set_wire_load_mode(2)

set_wire_load_selection_group(2)

remove_wire_load_model(2)

set_load(2)

set_local_link_library(2)

link_library(3)

set_wire_load_mode

Sets the **wire_load_model_mode** attribute on the current design, specifying how wire load models are to be used to calculate wire capacitance in nets.

SYNTAX

```
status set_wire_load_mode
      mode_name
```

Data Types

mode_name string

ARGUMENTS

mode_name

Specifies the value with which to set the **wire_load_model_mode** attribute, thus specifying the mode to be used to handle hierarchical wire load models. Allowed values are as follows:

- The value of *top* (the default) specifies that the wire capacitance of all nets is calculated using the wire load model set on the top-level design.
- The value of *enclosed* specifies that the wire capacitance of each net is calculated using the wire load model set on the smallest subdesign that completely encloses that net. You must specify this mode in order to use the **set_wire_load_min_block_size** commands.
- The value of *segmented* specifies that for each net that crosses hierarchical subdesigns, the wire capacitance is calculated for each segment of the net based on the wire load model set on the subdesign that contains that segment. The total wire capacitance of the net is the sum of the wire capacitances of its segments.

DESCRIPTION

The **set_wire_load_mode** command sets the **wire_load_model_mode** attribute on the current design, specifying how hierarchical wire load models are to be used to calculate wire capacitance of nets in the current design. If you use the **set_wire_load_min_block_size** command to set the **min_block_size** attribute and execute this command to set the *mode_name* to be different from *enclosed*, the **min_block_size** attribute is canceled.

The **report_design** command displays the wire loading model and mode for the current design, and the libraries to which the current design is linked.

To specify distinct wire load models for various hierarchical cells in the top-level design, specify wire load models on the designs

referenced by these hierarchical cells; wire load models are reference-specific and not instance-specific. You must also ensure that the **wire_load_model_mode** attribute of the topmost design is not *top*; otherwise, the wire load models selected for lower-level blocks in the design are ignored.

If you do not specify a **wire_load_model_mode** for a design, **compile** searches for a default **wire_load_model_mode** in the first library in the **link_path**. If that library does not have a default **wire_load_model_mode**, *top* is the default. If the mode for the top-level design is *top*, the wire load model on subdesigns and subclusters has no effect, and the top-level wire load model is used to compute wire capacitance for all hierarchical or top-level nets within a top-level design.

If the mode for the top-level design is either *enclosed* or *segmented*, wire load models on subdesigns and subclusters are used to calculate wire capacitance in nets inside these blocks. When the **auto_wire_load_selection** variable is *true*, if a wire load model is not specified for a subdesign, and if the first library in the **link_path** contains a **wire_load_selection** table, the **compile** command automatically selects a wire load model based on the cell area of the subdesign. If you do not define a **wire_load_selection** table, the wire load model of the parent design or cluster is used. If the top-level mode is *enclosed* or *segmented*, the wire load model of the subdesign is not set by **characterize** unless the subdesign does not have a wire load. When the **auto_wire_load_selection** variable is *false*, if a wire load model is not specified for a subdesign, no wire load model is selected.

If the model mode is *enclosed*, the wire load model of the smallest design that encloses a net completely is used to calculate the wire capacitance of that net. The number of pins on the net equals the number of leaf pins on the net.

EXAMPLES

The following example sets the **wire_load_model** attribute to *enclosed* for the top-level design named TOP:

```
prompt> current_design TOP
```

```
prompt> set_wire_load_model enclosed
```

SEE ALSO

- characterize(2)
- compile(2)
- current_design(2)
- remove_wire_load_model(2)
- report_lib(2)
- reset_design(2)
- set_local_link_library(2)
- set_wire_load_min_block_size(2)
- set_wire_load_model(2)
- set_wire_load_selection_group(2)
- auto_wire_load_selection(3)
- link_library(3)

set_wire_load_model

Sets the **wire_load_attach_name** attribute on designs, ports, hierarchical cells of current design, for selecting a wire load model to use in calculating wire capacitance.

SYNTAX

```
status set_wire_load_model
      -name model_name
      [-library lib]
      [-min] [-max]
      [object_list]
```

Data Types

model_name string
object_list collection

ARGUMENTS

-name *model_name*

Specifies the name of the wire load model to be used. The model must be defined in the library or in one of the libraries in the **link_library** (including the **local_link_library**). This is a required option.

-library *lib*

Specifies the library that contains the desired wire load model. This is either a library or a collection. The first library in the collection that contains the wire load model is selected.

-min

Indicates that the wire load model or selection group is to be used for minimum delay analysis only. You cannot use the **-min** option to set a different wire load mode or minimum block size, because both minimum and maximum delay analysis always use the same values for these parameters.

-max

Indicates that the wire load model or selection group is to be used for maximum delay analysis only. Any model set for minimum delay analysis is not affected.

object_list

Specifies a list of ports, designs, and/or cells, to which this wire load model is to be assigned. By default, the current design is assigned the wire load model.

DESCRIPTION

The **set_wire_load_model** command sets the **wire_load_attach_name** attribute on the specified ports, designs, or cells specified in the object list, or on the current design, for selecting a wire load model to use in calculating wire capacitance. Issuing the command without options specifies a wire load model for the top-level design. The **-name** option is required.

When a design is specified, this wire load model is applied to all nets in the design at the top level and those under the hierarchy. If cells are specified, they are searched in the current design. The wire load model set on a cell overrides the wire load model of the design for the cell.

You can use this command to set a port's external wire load model. The wire load model set on a port overrides the wire load model of the design for the net connected to that port.

If no wire load model is specified for a design, a default is selected. The tool searches the first library on the link path (or the first library in the design's **local_link_library**) for a default wire load model to use. When the **auto_wire_load_selection** attribute is set to true, if the library has wire_load_selection tables, the cell area of the current design is used to automatically select the appropriate wire load model from the appropriate wire_load_selection table. If the library has more than one table, it typically specifies that one of the tables is the default (by setting the **default_wire_selection_group** attribute). You can override this value by using the **set_wire_load_selection_group** command. If you did not set the minimum block size by using the **set_wire_load_min_block_size** command, the cell area of the design is assumed to be at least as large as the value specified for this option when selecting the appropriate wire load to use. If there are no wire_load_selection tables, a default_wire_load is used if there is one in the library. If the library has neither of the aforementioned, no wire load model is used.

When the **auto_wire_load_selection** attribute is set to false, and no *model_name* is specified, no wire load model is used.

The **report_design** command displays the wire load model and mode for the current design, and the libraries to which the current design is linked. The **report_wire_load** command provides a more detailed report on the wire load models associated with a design.

The **report_lib** command shows the wire load models that are defined in the specified library, as well as any wire load selection tables that are defined. You can use the **report_port** command to display the wire load model of the ports of a design.

You can specify instance-specific wire load models for hierarchical cells in the current design. The wire load models on the designs referenced by these hierarchical cells are not changed.

If you specify wire load model on subdesigns, you must ensure that the wire load mode of the top design is not **top**; otherwise, the wire load models selected for lower-level blocks in the design are ignored.

If you do not specify a wire load mode for a design, the tool searches for a default wire load mode in the first library in the link path during **compile**. If that library does not have a default wire load mode, **top** is assumed. You can override a mode by using the **set_wire_load_mode** command. For a definition of the **top**, **enclosed**, and **segmented** values of the **wire_load_model_mode** attribute, and an explanation of how a default wire load model is selected, see the man page for the **set_wire_load_mode** command.

Wire resistance and wire area are also calculated according to the rules for wire capacitance.

With the **-min** option, you can use the **set_wire_load_model** command to specify a different wire load model to use for minimum delay analysis. Unless explicitly specified, minimum delay analysis uses the same wire loads that maximum delay analysis uses. You can set different wire loads for minimum delay analysis on designs or ports. You cannot specify a different wire load mode or a different minimum block size for minimum delay analysis only. In these cases these same value is always used for both minimum and maximum analysis.

Wire area is always calculated by using the same wire loads used for maximum delay analysis.

Wire loading models contain all the information required by **compile** to estimate interconnect wiring delays. The following is an example of a wire loading model definition, as it appears in the source text of a library.

```
wire_load("90x90") {
  resistance : 0 ;
  capacitance : 1 ;
  area : 0 ;
  slope : 1.64 ;
  fanout_length( 1 , 1.9 ) ;
  fanout_length( 2 , 2.8 ) ;
  fanout_length( 3 , 4.7 ) ;
}
```

The name of this model is 90x90. The fields in this model are defined as follows:

resistance

The resistance per unit length of interconnect wire.

capacitance

The capacitance per unit length of interconnect wire.

area

The net-area per unit length of interconnect wire.

fanout_length (fanout1, length1)

These pairs of numbers specify a lookup table that the tools use to estimate the length of interconnect wire being driven by a cell with a given fanout. You must specify at least one of these pairs (with fanout = 1), and can specify as many additional pairs as necessary to characterize the desired fanout length behavior. Linear interpolation between the two nearest pairs is used to estimate any points that are not in the lookup table.

slope

This value is used to characterize linear fanout or length behavior beyond the last pair specified in the lookup table.

To remove a wire load model from a design, use the **remove_wire_load_model** command, or use the **reset_design** command.

EXAMPLES

The following example sets the wire load model on the top-level design, on which there is no **local_link_library** set:

```
prompt> report_lib tech_lib
```

```
.
.
.
```

Wire Loading Model:

```
Name Library Res Cap Area Slope Fanout Length
-----
05x05 tech_lib 0.0000 1.0000 0.0000 0.1860 10 0.3900
10x10 tech_lib 0.0000 1.0000 0.0000 0.3110 1 0.5300
```

```
.
.
.
```

```
prompt> link_library = {tech_lib.db}
```

```
prompt> set_wire_load_model -name "10x10"
```

In the following example, the **-library** option specifies a wire load model of 10x10 from the my_lib.db library:

```
prompt> set_wire_load_model -name "10x10" -library my_lib.db
```

The following example sets the 10x10 wire load model on the subdesign named LOW to be used for maximum and minimum delay analysis. Then, the wire load model is set to 20x20 and the wire load model mode is set to **enclosed** for the top-level design TOP. Finally, a different wire load model named 20x20MIN is set at the top level to be used for minimum delay analysis only.

```
prompt> current_design LOW
prompt> set_wire_load_model -name "10x10"
prompt> current_design TOP
prompt> set_wire_load_mode enclosed
prompt> set_wire_load_model -name "20x20"
prompt> set_wire_load_model -name "20x20MIN" -min
```

The following example sets no wire load model on the LOW subdesign, leaving it to be selected based on its cell area. Then, the wire load model is set to 20x20 and the wire load model mode is set to **enclosed** for the TOP top-level design, with a **-min_block_size** option specifying to assume a block size of at least 200.0 for all subdesigns (in this case, LOW).

```
prompt> current_design LOW /* no wire-load set now */
prompt> current_design TOP
prompt> set_wire_load_mode enclosed
prompt> set_wire_load_min_block_size 200.0
prompt> set_wire_load_model -name "20x20"
```

The following sequence of commands describes the external fanout of a port:

```
prompt> set_wire_load_model [get_ports O1] "default_wl"
prompt> set_port_fanout_number 5 [get_ports O1]
```

Assume U1, U2, and U3 are 3 cells in the design named TOP, and all of them are instances of a subdesign named LOW. The following commands set the 10x10 wire load model on LOW and the 20x20 wire load model on U1 and U2:

```
prompt> current_design TOP
prompt> set_wire_load_model -name "10x10" LOW
prompt> set_wire_load_model -name "20x20" {U1 U2}
```

SEE ALSO

```
characterize(2)
current_design(2)
remove_wire_load_model(2)
report_lib(2)
reset_design(2)
set_load(2)
set_local_link_library(2)
set_port_fanout_number(2)
set_wire_load_min_block_size(2)
set_wire_load_mode(2)
set_wire_load_selection_group(2)
auto_wire_load_selection(3)
link_library(3)
```

set_wire_load_selection_group

Specify a selection group to use for determining a wire load model to be assigned to designs and cells or to a specified cluster.

SYNTAX

```
int set_wire_load_selection_group
    [-library lib] [-min]
    [-max]
    group_name [object_list]
```

Data Types

```
group_name    string
object_list   list
```

ARGUMENTS

-library *lib*

Specifies the library from which to select the wire load model. This is either a library name or a collection. The first library in the collection that contains the wire load model is selected.

-min

Specifies to use the selection group for minimum delay analysis only. You cannot use the **-min** option to set a different wire load mode or minimum block size, because both minimum and maximum delay analysis always use the same values for these parameters.

-max

Specifies to use the selection group for maximum delay analysis. Any model set for minimum delay analysis is not affected.

group_name

Specifies the name of the selection group to use to determine the wire load model based on area.

object_list

Specifies a list of designs and/or cells, to which the wire load model selection group attributes are to be assigned. If the *object_list* is not specified, the current design is assigned the attributes.

DESCRIPTION

This command sets the **wire_load_model_selection_group** attribute to *group_name*, and the **wire_load_model_selection_group_lib** attribute to the first library in *lib* that contains the group, on the designs and cells specified in *object_list* or on the specified cluster of current design. If none of them is specified, current design is assigned the attributes. You can define a selection group for designs, cells, or a cluster, but not for ports.

The area of the specified selection group is used to determine the wire load model for designs and cells, or cluster. It is supported only for the *enclosed* wire load model mode, which you specify using the **set_wire_load_mode** command. This command has an effect only when the variable **auto_wire_load_selection** is *true*.

To view the wire load associated with a specific cluster, use the **report_clusters** command. For a more detailed report on the wire load models associated with either a design or a cluster, use the **report_wire_load** command. To view the wire loading models defined in a library, as well as any *wire_load_selection* tables that might be defined there, use the **report_lib** command.

With the **-min** option, you can use this command to specify a wire load selection group to use for minimum delay analysis. Unless explicitly specified, minimum delay analysis uses the same wire loads as does maximum delay analysis. Different wire loads for minimum delay analysis can be set on designs, cells, or clusters. You cannot specify a different wire load mode or a different minimum block size for minimum delay analysis only. In these cases, the same value is always used for both minimum and maximum analysis.

Wire area is always calculated based on the wire loads used for maximum delay analysis.

EXAMPLES

The following example sets the selection group for a sub-design LOW and a hierarchical cell U1/U2 in the top design.

```
prompt> current_design TOP
```

```
prompt> set_wire_load_selection_group 2layermetal {LOW U1/U2}
```

The following example sets the selection group for the current design.

```
prompt> current_design TOP
```

```
prompt> set_wire_load_selection_group 2layermetal
```

SEE ALSO

- `characterize(2)`
- `current_design(2)`
- `set_load(2)`
- `remove_wire_load_model(2)`
- `report_lib(2)`
- `reset_design(2)`
- `set_local_link_library(2)`
- `set_wire_load_min_block_size(2)`
- `set_wire_load_mode(2)`
- `set_wire_load_model(2)`
- `link_library(3)`

set_wrapper_configuration

Sets the default wrapper configuration for the current design.

SYNTAX

```
status set_wrapper_configuration
  -class core_wrapper | shadow_wrapper
  [-style dedicated | shared]
  [-core core_list]
  [-dedicated_cell_type WC_D1 | WC_D1_S | none]
  [-shared_cell_type WC_S1 | WC_S1_S | none]
  [-dedicated_design_name design_name]
  [-shared_design_name design_name]
  [-register_io_implementation swap | in_place]
  [-use_dedicated_wrapper_clock true | false]
  [-safe_state 0 | 1 | none]
  [-delay_test true | false]
  [-max_length integer]
  [-chain_count integer]
  [-mix_cells true | false]
  [-input_shift_enable port_name]
  [-output_shift_enable port_name]
  [-maximize_reuse enable | disable]
  [-reuse_threshold threshold_value]
  [-depth_threshold threshold_value]
  [-use_system_clock_for_dedicated_wrp_cells enable | disable]
  [-end_of_chain_logic enable | disable]
  [-input_mask_bound mask_bound]
  [-output_mask_bound mask_bound]
  [-hier_wrapping enable | disable]
  [-use_separate_wrapper_chain_controls enable | disable]
  [-shift_enable port_name]
  [-test_mode mode_name]
  [-create_only_wrapper_modes mode_list]
  [-add_wrapper_cells_to_power_domains enable | disable]
  [-gate_dedicated_wrapper_cell_clk enable | disable]
  [-gate_cells none | existing_cg | all]
  [-hold_mux_for_shared_wrapper_cells enable | disable]
  [-no_dedicated_wrapper_cells enable | disable]
  [-feedthrough_chains enable | disable]
  [-input_wrapper_cells cell_list]
  [-output_wrapper_cells cell_list]
  [-force_clock_enable all]
```

Data Types

core_list list

design_name string
integer threshold_value
port_name string
threshold_value integer
mask_bound pin or port
mode_name string
mode_list list
cell_list list

ARGUMENTS

-class *core_wrapper* | *shadow_wrapper*

Specifies the name of the class to which the wrapper configuration applies.

This option is required.

-style *dedicated* | *shared*

Specifies the wrapper cell style to be used for core-wrapping the current design.

The default is **dedicated** in the simple and delay test wrapper flows; the default is **shared** in the maximized reuse flow.

-core *core_list*

Specifies the list of core cells to which the configuration applies.

The default is to apply the configuration to the current design, rather than applying it to specific cells.

-dedicated_cell_type *WC_D1* | *WC_D1_S* | *none*

Specifies the cell type to be used for dedicated wrapper cells when core-wrapping the current design.

The default is **WC_D1** for cells without a SAFE value and **WC_D1_S** for cells with a SAFE value.

-shared_cell_type *WC_S1* | *WC_S1_S* | *none*

Specifies the cell type to be used for shared wrapper cells when core-wrapping the current design.

The default is **WC_S1** for cells without a SAFE value and **WC_S1_S** for cells with a SAFE value.

-dedicated_design_name *design_name*

Specifies the name of the wrapper cell design to be used for dedicated wrapper cells when core-wrapping the current design.

The specified design must be previously specified with the **define_dft_design** command.

There is no default for this option.

-shared_design_name *design_name*

Specifies the name of the wrapper cell design to be used for shared wrapper cells when core-wrapping the current design.

The specified design must be previously specified with the **define_dft_design** command.

There is no default for this option.

-register_io_implementation *swap* | *in_place*

Specifies the implementation to be used for synthesizing shared wrapper cells when core-wrapping the current design.

The default is **swap** in the simple and delay test wrapper flows; the default is **in_place** in the maximized reuse flow.

-use_dedicated_wrapper_clock true | false

Specifies if a dedicated wrapper clock should be used for shared wrapper cells added by core wrapping.

The default is **false**.

-safe_state 0 | 1 | none

Specifies the safe state value to be used for wrapper cells added by core wrapping.

The default is **none**.

-delay_test true | false

Specifies if additional wrapper test modes that are used for delay testing should be created by core wrapping. In these additional wrapper test modes, all wrapper chains contain either input wrapper cells or output wrapper cells. Wrapper chains containing input wrapper cells and wrapper chains containing output wrapper cells are controlled by different wrapper shift enables.

The default is **false**.

-max_length integer

Specifies the maximum length of a wrapper chain in all wrapper modes. Wrapper chain lengths in scan compression modes are not controlled by this option.

The default is that wrapper chain length is not controlled this option.

-chain_count integer

Specifies the maximum number of wrapper chains in all wrapper modes. The number of wrapper chains in scan compression modes are not controlled by this option.

There is no default for this option; it has no effect unless explicitly set.

When you do not use either the **-max_length** or **-chain_count** option, the wrapper chain length or number of wrapper chains are controlled by the **-max_length** and **-chain_count** options of the **set_scan_configuration** command.

-mix_cells true | false

Specifies if wrapper cells for input or output ports can be mixed in a wrapper chain.

If this option is set to **false**, all wrapper cells in a wrapper chain are either associated with input ports or output ports but not both.

If this option is set to **true**, input and output wrapper cells can be mixed in the same wrapper chain.

The default is **true** in the simple and delay test wrapper flows; the default is **false** in the maximized reuse flow.

-input_shift_enable port_name

Specifies the wrapper shift enable port for wrapper chains that contain only wrapper cells for input ports.

The *port_name* argument must be defined as a **wrp_shift** DFT signal.

There is no default for this option.

-output_shift_enable port_name

Specifies the wrapper shift enable port for wrapper chains that contain only wrapper cells for output ports.

The *port_name* argument must be defined as a **wrp_shift** DFT signal.

There is no default for this option.

-maximize_reuse enable | disable

Specifies that the tool can reuse I/O registers in the fanin or fanout cone of ports as shared wrapper cells. These I/O registers do not need to be on the sensitive path of the ports; any combinational logic is allowed between the ports and the I/O registers.

Dedicated wrapper cells are not added to ports that are not register bounded.

Dedicated wrapper cells are added to ports that are connected to clock gating cells or registers inside a DFT model.

Currently the tool expects to see the netlist of a DFT model in addition to the model as the model does not show the complete association of pins to the registers inside the model. If netlist is not loaded for a model cell, the tool assumes that the model has only combinational logic.

When a dedicated wrapper cell is added to a port that is connected to I/O registers, a user-specified clock is used for the dedicated wrapper cell. In the absence of such a specification, the most used clock domain of I/O registers is used as the clock of the dedicated wrapper cell. If no such clock is found, a dedicated wrapper clock is used.

If an I/O register is common to both an input port and a output port, the I/O register is classified as an input I/O register.

These I/O registers are also used as normal scan cells in *internal_scan* mode.

When the **-maximize_reuse** option is enabled, the tool sets the following options automatically:

- style shared
- register_io_implementation in_place
- mix_cells false
- use_system_clock_for_dedicated_wrp_cells enable

The default of this option is **disable**.

-reuse_threshold threshold_value

Specifies the maximum number of I/O registers that can be reused as shared wrapper cells in the maximized reuse flow.

This option is applicable only when the **-maximize_reuse** option is set to **enable**.

When the number of I/O registers detected for a port exceeds the specified threshold value, a dedicated wrapper cell is added to the port.

The default is 1. If the option is set to 0, all I/O registers associated with a port are reused as shared wrapper cells.

-depth_threshold threshold_value

Specifies the maximum combinational depth of I/O registers that can be reused as shared wrapper cells for a port in the maximized reuse flow.

This option is applicable only when the **-maximize_reuse** option is set to **enable**.

The combinational logic depth is defined as the maximum number of combinational logic levels, including buffers and inverters, between a port and an associated I/O register. For a given port, if the combinational logic depth of any associated I/O register exceeds this value, a dedicated wrapper cell is added to the port.

There is no default for this option. When not specified, the check is not performed and all I/O registers associated with a port are reused as shared wrapper cells, subjective to other checks.

-use_system_clock_for_dedicated_wrp_cells enable | disable

Controls whether the tool uses the system clock of I/O registers of the port as the clock for the dedicated wrapper cell.

When a dedicated wrapper cell is added to a port and this option is enabled, the tool uses the system clock of I/O registers of the port as the clock for the dedicated wrapper cell.

If the port is connected to a single I/O register and the tool adds a dedicated wrapper cell to the port, the clock of the associated I/O register is used as the clock for the dedicated wrapper cell.

The default wrapper clock is used for a dedicated wrapper cell if the associated port is not connected to any I/O registers.

The tool issue an error if more than one I/O register is found for a port and all of these I/O registers do not use the same clock domain and you did not specify any clock for the port. The command is terminated in such a case.

If you specify a clock for such a port, the tool treats the error as a warning and uses the specified clock as the clock for the dedicated wrapper cell.

The default is **disable** in the simple and delay test wrapper flows; the default is **enable** in the maximized reuse flow.

-end_of_chain_logic enable | disable

Controls whether the tool adds end-of-chain logic to all wrapper chains of all modes. Here is the description of the logic added to wrapper chains when this option is enabled.

All wrapper chains are driven with the following logic:

$$\text{masked\_wsi} = (\sim\text{wse} \ \& \ \text{wrapper\_mode} \ \& \ \text{input\_mask\_bound}) \mid \text{wsi}$$

The following logic is added to all wrapper chain outputs:

$$\text{masked\_wso} = ((\sim\text{wse} \ \& \ \text{wrapper\_mode} \ \& \ \text{input\_mask\_bound}) \mid \text{output\_mask\_bound}) \mid \text{wso}$$

where

wse - wrapper chain scan enable
 wrapper_mode - the test mode in which the wrapper chain is active
 wsi - wrapper chain's original scan input
 masked_wsi - wrapper chain's masked(new) scan input
 wso - wrapper chain's original scan output
 masked_wso - wrapper chain's masked(new) scan output

The default is **disable** so that no end-of-chain logic is added to wrapper chains.

-input_mask_bound mask_bound

Specifies the mask bound to be used in the end-of-chain logic added by the tool for wrapper chain inputs.

The logic added is described in the **-end_of_chain_logic** description. The mask bound can be a pin or a port.

This option is effective only if the **-end_of_chain_logic** option is enabled.

There is no default for this option.

-output_mask_bound mask_bound

Specifies the mask bound to be used in the end-of-chain logic added by the tool for wrapper chain outputs.

The logic added is described in the **-end_of_chain_logic** description. The mask bound can be a pin or a port.

This option is effective only if the **-end_of_chain_logic** option is enabled.

There is no default for this option.

-hier_wrapping enable | disable

Enables or disables hierarchical wrapping.

Hierarchical wrapping is supported only when the **-maximize_reuse** option is enabled and there is only one scan mode synthesized in DFT insertion.

When hierarchical wrapping is enabled, the following core wrapping functionality is used:

All input wrapper cells are stitched into one or more input wrapper chains, controlled by an input shift-enable signal.

All output wrapper cells are stitched into one or more output wrapper chains, controlled by an output shift-enable signal.

Separate input and output control signals are used for the following wrapper control signal types:

- Wrapper shift signal
- Wrapper capture signal (if the input or output wrapper chains include a dedicated wrapper cell)
- SAFE mode enable signal (if the input or output wrapper chains include a wrapper cell that drives a SAFE value)

There is no wrapper mode logic added to the design.

The default is **disable**.

-use_separate_wrapper_chain_controls enable | disable

Specifies whether to use separate control signals for each wrapper chain.

This option is used in the hierarchical wrapping flow to create cores with increased wrapper chain reclassification flexibility. Hierarchical wrapping is enabled by the **-hier_wrapping** option.

When this option is set to the default of **disable**, the tool uses the minimum set of wrapper control signals needed for all wrapper chains.

When this option is set to **enable**, the tool uses separate signals for each individual input and output wrapper chain for the following wrapper control signal types:

- Wrapper shift signal
- Wrapper capture signal (if the wrapper chain includes a dedicated wrapper cell)
- SAFE mode enable signal (if the wrapper chain includes a wrapper cell that drives a SAFE value)

When you use separate control signals for each wrapper chain, the tool can independently reclassify each wrapper chain when the core is integrated at a higher level. This can be useful when some wrapper chains communicate with logic within the integration level (so that the wrapper chain could be reclassified as a standard scan chain) and some communicate with logic outside the integration level (so that the wrapper chain should remain as a wrapper chain).

This option has no effect when the **-hier_wrapping** option is set to its default of **disable**. For more information about wrapper chain classification, see the description for the **-hier_wrapping** option.

-shift_enable *port_name*

Specifies the wrapper shift enable port for wrapper chains.

This option is applicable only when the **-mix_cells** option is set to **true**.

The *port_name* argument must be defined as a **wrp_shift** DFT signal.

If the option is not specified, the tool uses one of the wrapper shift signals. If no wrapper shift signal is specified, the tool creates a wrapper shift signal.

-test_mode *mode_name*

Specifies the test mode to which the specification applies.

Only the following options can be specified with this option when the value of the option is not **all**:

-max_length
-chain_count
-mix_cells
-input_shift_enable
-output_shift_enable
-shift_enable

The values of all other options are taken from the wrapper configuration specification for the **all** mode.

The default is **all** (the specification applies to all modes).

-create_only_wrapper_modes *mode_list*

Specifies that only wrapper test modes with the specified usages should be created.

This option is used for analog/mixed-signal designs that do not require any inward-facing wrapper modes. By default, the tool creates both inward-facing and outward-facing wrapper modes. Valid argument values to restrict mode creation are:

- {wrp_of} - Creates only outward-facing wrapper modes. This provides only outward-facing wrapper functionality for analog/mixed-signals cores.
- {wrp_of wrp_safe} - Creates both outward-facing wrapper modes and safe wrapper modes. This can be used to also create a safe mode for the block that drives constant output values with no scan requirements.

The default is an empty list ({}), which deactivates the option so that the tool creates both inward-facing and outward-facing wrapper modes.

-add_wrapper_cells_to_power_domains enable | disable

Enables or disables the addition of dedicated wrapper cells to associated power domain hierarchies.

This option has no effect if there are no power domains in the design or no dedicated wrapper cells are added during wrapper insertion.

When this option is enabled, the tool identifies the hierarchy in which a dedicated wrapper cell is added by using the following method.

If the port of the dedicated wrapper cell is connected to an I/O register, the top-level hierarchy of the power domain of the I/O register is used as the location for the dedicated wrapper cell.

If no I/O register is found for the port of the dedicated wrapper cell, the tool checks if the port is connected to a CTL modeled instance. If so, the top-level hierarchy of the power domain of the CTL modeled instance is used as the location for the dedicated wrapper cell.

If neither I/O register nor CTL modeled instance is found to be connected to the port of the dedicated wrapper cell, the top-level hierarchy of the first connected gate is used as the location for the dedicated wrapper cell.

If the I/O register or CTL modeled instance or the gate connected to the port of the dedicated wrapper cell does not have a power domain, the dedicated wrapper cell is added to the top-level hierarchy.

When this option is enabled, no additional hierarchy is created for dedicated wrapper cells added to the design.

When this option is enabled, any user-specified wrapper logic location (**set_dft_location ...**) is ignored.

The default is **disable** so that dedicated wrapper cells are added to the user-specified wrapper logic location if specified (**set_dft_location ...**) or to the top-level hierarchy.

-gate_dedicated_wrapper_cell_clk enable | disable

Enables or disables gating of the clock of all dedicated wrapper cells synthesized during wrapper insertion.

When this option is enabled, the clock of all dedicated wrapper cells is gated with simple combinational logic such that the clock is off in all test modes in which dedicated wrapper cell is not active.

For example, the clock of dedicated wrapper cells is off in `mission_mode` and `internal_scan` test modes.

The default is **disable** so that the clock of the dedicated wrapper cells is not gated.

-gate_cells none | existing_cg | all

Controls whether clock-gating logic is inserted or modified to reduce power consumption in core wrapping modes.

This feature implements the following behavior for integrated clock-gating cells:

- In EXTEST (outward-facing) modes, clocks to core register cells are disabled.
- In SAFE mode, clocks to both core register cells and wrapper cells are disabled.

When this option is set to **existing_cg**, the tool modifies only existing integrated clock-gating cells. Both the functional and test enable pin connections are modified.

When this option is set to **all**, the tool modifies existing integrated clock-gating cells, and it also inserts integrated clock-gating cells for more aggressive power reduction. (You must set the **test_icg_p_ref_for_dft** and **test_icg_n_ref_for_dft** application variables to specify the rising-edge and falling-edge ICG library cells to insert, respectively.)

The default is **none**, which disables the feature.

-hold_mux_for_shared_wrapper_cells enable | disable

Enables or disables a hold MUX for shared wrapper cells when the **-maximize_reuse** option is enabled.

When a hold MUX is added, shared wrapper cells hold the state when the cell is not in shift mode in test operation.

The default is not to add a hold MUX to shared wrapper cells.

This option has no effect when the **-maximize_reuse** option is disabled.

-no_dedicated_wrapper_cells enable | disable

When this option is set to **enable**, dedicated wrapper cells are not added to chains for the specified wrapper test mode.

The default is **disable** so that dedicated wrapper cells are added to chains for the specified wrapper test mode.

-feedthrough_chains enable | disable

Specifies whether feedthrough chains are created in transparent modes.

When this option is set to **disable**, feedthrough chains are not created in transparent modes. In this case, you no longer need to use dedicated wrapper scan I/Os for the core's outward-facing modes. The resulting core contains no feedthrough chains, and it can be integrated in top-level designs that do not have a top-level codec.

The default is **enable**, which creates feedthrough chains in transparent modes. This ensures that the core can be integrated in top-level designs that have a top-level codec.

This option affects only transparent test modes, which are defined with the **-transparent_mode_of** option of the **define_test_mode** command.

-input_wrapper_cells cell_list

Specifies additional scan cells to be classified as output-related registers to include in the input wrapper chain. The input is accepted as a simple list. Wildcards and collections are not supported.

This option is not cumulative; new specifications overwrite previous specifications. Specify an empty list to remove a previous specification.

This option is applicable only when the **-maximize_reuse** option is set to **enable**.

-output_wrapper_cells *cell_list*

Specifies additional scan cells to be classified as input-related registers to include in the output wrapper chain. The input is accepted as a simple list. Wildcards and collections are not supported.

This option is not cumulative; new specifications overwrite previous specifications. Specify an empty list to remove a previous specification.

This option is applicable only when the **-maximize_reuse** option is set to **enable**.

DESCRIPTION

This command specifies the default configuration to be used by the core wrapper or shadow wrapper application.

Use the **reset_wrapper_configuration** command to reset the current wrapper configuration of the design.

EXAMPLES

The following example specifies that the WC_D1_S cell is to be used as dedicated wrapper cell, WC_S1_S to be used as shared wrapper cell with safe value 0 in core wrapping.

```
prompt> set_wrapper_configuration -class core_wrapper \
      -dedicated_cell_type WC_D1_S -shared_cell_type WC_S1_S \
      -safe_state 0
```

Accepted wrapper configuration for design '...' for mode all_dft.

1

```
prompt> set_wrapper_configuration -class core_wrapper \
      -style shared -register_io_implementation in_place -mix_cells false \
      -maximize_reuse enable -reuse_threshold 4
```

Accepted wrapper configuration for design '...' for mode all_dft.

1

```
prompt> define_test_mode WINTTEST -usage wrp_if
Defining test mode WINTTEST in view existing_dft
Defining test mode WINTTEST in view spec
Current test mode is 'WINTTEST'.
```

1

```
prompt> define_test_mode WEXTTEST -usage wrp_of
Defining test mode WEXTTEST in view existing_dft
Defining test mode WEXTTEST in view spec
Current test mode is 'WEXTTEST'.
```

1

```
prompt> set_wrapper_configuration -class core_wrapper \
      -test_mode all -maximize_reuse enable \
      -reuse_threshold 4 -mix_cells false
```

Accepted wrapper configuration for design '...' for mode all_dft.

1

```
prompt> set_wrapper_configuration -class core_wrapper \
```

```
-test_mode WINTEST -chain_count 4
Accepted wrapper configuration for design '...' for mode WINTEST.
1
prompt> set_wrapper_configuration -class core_wrapper \
-test_mode WEXTTEST -mix_cells true -chain_count 1
Accepted wrapper configuration for design '...' for mode WEXTTEST.
1
```

SEE ALSO

insert_dft(2)
preview_dft(2)
reset_wrapper_configuration(2)
report_wrapper_configuration(2)
set_boundary_cell(2)

set_zero_interconnect_delay_mode

Forces the timer to ignore the contribution on a timing path from any wire capacitance in the design.

SYNTAX

status **set_zero_interconnect_delay_mode**
[true | false]

ARGUMENTS

true | false

Enables the zero interconnect delay mode when set to **true**. Disables the zero interconnect delay mode when set to **false**. If no value is specified, the default is **true**.

DESCRIPTION

This command enables or disables the zero interconnect delay mode. The zero interconnect delay mode forces the timer to ignore contributions on a timing path from any wire capacitance in a design. It forces all wire capacitance to be as trivial as 0, regardless of the wire load model used in the design or any back-annotated capacitance on a wire. Disabling the mode allows the timer to consider the wire capacitance as it did before enabling the mode.

The command is used primarily in preplacement and postplacement steps to assess design and constraint feasibility. When the mode is enabled, the design is analyzed with only cell delay and the capacitance of the pin load on all wires in the design to determine if the design can meet timing goals. It also helps when debugging potential missing timing exceptions in the constraints. Zero interconnect delay mode must not be used in the final implementation step.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

get_zero_interconnect_delay_mode(2)
report_timing(2)

setenv

Sets the value of a system environment variable.

SYNTAX

```
string setenv  
      variable_name new_value
```

Data Types

```
variable_name  string  
new_value      string
```

ARGUMENTS

variable_name

Names of the system environment variable to set.

new_value

Specifies the new value for the system environment variable.

DESCRIPTION

The **setenv** command sets the specified system environment *variable_name* to the *new_value* within the application. If the variable is not defined in the environment, the environment variable is created. The **setenv** command returns the new value of *variable_name*. To develop scripts that interact with the invoking shell, use **getenv** and **setenv**.

Environment variables are stored in the Tcl array variable **env**. The environment commands **getenv**, **setenv**, and **printenv** are convenience functions to interact with this array.

The **setenv** command sets the value of a variable only within the process of your current application. Child processes initiated from the application using the **exec** command after a usage of **setenv** inherit the new variable value. However, these new values are not exported to the parent process. Further, if you set an environment variable using the appropriate system command in a shell you invoke using the **exec** command, that value is not reflected in the current application.

EXAMPLES

The following example changes the default printer.

```
prompt> getenv PRINTER  
laser1  
prompt> setenv PRINTER "laser3"  
laser3  
prompt> getenv PRINTER  
laser3
```

SEE ALSO

- [exec\(2\)](#)
- [getenv\(2\)](#)
- [printenv\(2\)](#)
- [printvar\(2\)](#)
- [set\(2\)](#)
- [sh\(2\)](#)
- [unset\(2\)](#)

sh

Executes a command in a child process.

SYNTAX

string **sh**

ARGUMENTS

The **sh** command has no arguments.

DESCRIPTION

This is very similar to the **exec** command. However, file name expansion is performed on the arguments. Remember that quoting and grouping is in terms of Tcl. Arguments which contain spaces will need to be grouped with double quotes or curly braces. Tcl special characters which are being passed to system commands will need to be quoted and/or escaped for Tcl. See the examples below.

EXAMPLES

This example shows how you can remove files with a wildcard.

```
prompt> ls aaa*
aaa1  aaa2  aaa3
prompt> sh rm aaa*
prompt> ls aaa*
Error: aaa*: No such file or directory
    Use error_info for more info. (CMD-013)
```

This example shows how to grep some files for a regular expression which contains spaces and Tcl special characters:

```
prompt> exec cat test3.out
blah blah blah
blah blah blah c blah
input [1:0] A;
```

```
output [2:0] B;  
prompt>  
prompt> sh egrep -v {[ ]+c[ ]+} test3.out  
blah blah blah  
prompt>  
prompt> sh egrep {t[ ]+\\} test3.out  
input [1:0] A;  
output [2:0] B;
```

SEE ALSO

exec(2)

sh_list_key_bindings

Displays all the key bindings and edit mode of current shell session. This variable is for use in Tcl mode only.

SYNTAX

```
int sh_list_key_bindings [-nosplit]
```

ARGUMENTS

-nosplit

Indicates that lines are not to be split when column fields overflow.

DESCRIPTION

The **sh_list_key_bindings** command displays current key bindings and the edit mode. To change the edit mode, command **set_cle_options** can be used on the shell.

The text CTRL+K is read as `Control+K' and describes the character produced when the k key is hit while the Control key is depressed.

The text META+K is read as `Meta+K' and describes the character produced when the Meta key is depressed, and the k key is hit. The Meta key is labeled ALT on many keyboards. On keyboards with two keys labeled ALT (usually to either side of the space bar), the ALT on the left side is generally set to work as a Meta key. The ALT key on the right may also be configured to work as a Meta key or may be configured as some other modifier, such as a Compose key for typing accented characters.

If you do not have a Meta or ALT key, or another key working as a Meta key, the identical keystroke can be generated by typing ESC first, and then typing k. Either process is known as metafying the k key.

In vi mode, typing ESC will change the edit mode to alternate (command) mode. Alternate key bindings work only in vi alternate (command) mode.

In vi mode, if arrow keys are used while in input mode then the mode will be automatically changed to alternate (command). Arrow keys should not be used to move around in the line while in input mode. If there is any mistake has been made while entering text in input mode, first press ESC to return to alternate mode and then relocate the cursor to the position where the error was made.

EXAMPLES

SEE ALSO

`sh_enable_line_editing(3)`
`set_cle_options(2)`

shell_is_dcnxt_shell

Determines if the shell was invoked in DCNXT mode.

SYNTAX

status **shell_is_dcnxt_shell**

ARGUMENTS

The **shell_is_dcnxt_shell** command has no arguments.

DESCRIPTION

This command is used to check if the shell is in DCNXT mode. The command returns 1 if the shell is in DCNXT mode, or a 0 otherwise.

EXAMPLES

The following example shows how to check if shell was invoked in DCNXT:

```
prompt> if { [shell_is_dcnxt_shell] } { \  
    echo "You invoked dc_shell in DCNXT mode" \  
} else { \  
    echo "You are not in DCNXT mode" \  
}
```

shell_is_in_exploration_mode

Determines if the shell was invoked in DC Explorer.

SYNTAX

status **shell_is_in_exploration_mode**

ARGUMENTS

The **shell_is_in_exploration_mode** command has no arguments.

DESCRIPTION

This command is used to check if the shell is in DC Explorer. The command returns 1 if the shell is in DC Explorer, or a 0 otherwise.

EXAMPLES

The following example shows how to check if shell was invoked in DC Explorer:

```
prompt> if { [shell_is_in_exploration_mode] } { \  
    echo "You invoked de_shell in DC Explorer" \  
} else { \  
    echo "You are not in DC Explorer" \  
}
```

shell_is_in_ndm_mode

Determines if the Design Compiler shell is in NDM mode.

SYNTAX

status **shell_is_in_ndm_mode**

ARGUMENTS

The **shell_is_in_ndm_mode** command has no arguments.

DESCRIPTION

This command is used to check if the shell is in NDM mode. The command returns 1 if the shell is in NDM mode or 0 if it is in Milkyway mode

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to check the mode in which the shell has been invoked:

```
prompt> if { [shell_is_in_ndm_mode] } { \  
    echo "You invoked dc_shell in NDM mode" \  
} else { \  
    echo "You invoked dc_shell in Milkyway mode" \  
}
```

shell_is_in_topographical_mode

Determines if the Design Compiler shell was invoked in topographical mode.

SYNTAX

status **shell_is_in_topographical_mode**

ARGUMENTS

The **shell_is_in_topographical_mode** command has no arguments.

DESCRIPTION

This command is used to check if the shell is in topographical mode. The command returns 1 if the shell is in topographical mode, or a 0 otherwise.

EXAMPLES

The following example shows how to check the mode in which the shell has been invoked:

```
prompt> if { [shell_is_in_topographical_mode] } { \  
    echo "You invoked dc_shell in Topographical mode" \  
} else { \  
    echo "You invoked dc_shell in normal mode" \  
}
```

SEE ALSO

dc_shell(1)

shell_is_in_xg_mode

Determines if the shell is in XG mode.

SYNTAX

status **shell_is_in_xg_mode**

ARGUMENTS

The **shell_is_in_xg_mode** command has no arguments.

DESCRIPTION

This command is used to check if the shell is in XG mode. The command returns 1 if shell is in XG mode, or 0 otherwise. For use in Tcl-based dc_shell only. Since XG is the only mode, this command will be obsoleted in the future.

EXAMPLES

The following example shows how to determine the mode in which the shell has been invoked:

```
prompt> if { [shell_is_in_xg_mode] } { \  
    echo "You invoked dc_shell in XG mode" \  
} else { \  
    echo "You invoked dc_shell in normal mode" \  
}
```

sim_assertion_control

This is a simulation command.

SYNTAX

```
string sim_assertion_control
  [-elements element_list]
  [-exclude_elements exclude_list]
  [-domain domain_name]
  [-model model_name]
  [-controlling_domain controlling_domain_name]
  [-control_expr boolean_expression]
  [-type type]
  [-transitive transitive]
```

Data Types

<i>element_list</i>	set
<i>exclude_list</i>	set
<i>domain_name</i>	string
<i>model_name</i>	string
<i>controlling_domain_name</i>	string
<i>type</i>	string
<i>transitive</i>	Boolean

ARGUMENTS

Synthesis tool does not evaluate the arguments.

DESCRIPTION

This is a simulation command. It is ignored as it does not have functional impact in the synthesis tool. **save_upf** will not write out this command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

SEE ALSO

sim_corruption_control

This is a simulation command.

SYNTAX

```
string sim_corruption_control
  [-type type]
  [-elements element_list]
  [-exclude_elements exclude_list]
  [-model model_name]
  [-domain domain_name]
  [-transitive transitiveFP]
```

Data Types

```
type          string
element_list  set
exclude_list  set
model_name    string
domain_name   string
```

ARGUMENTS

Synthesis tool does not evaluate the arguments.

DESCRIPTION

This is a simulation command. It is ignored as it does not have functional impact in the synthesis tool. **save_upf** will not write out this command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

SEE ALSO

sim_replay_control

This is a simulation command.

SYNTAX

```
string sim_replay_control
  [-elements element_list]
  [-exclude_elements exclude_list]
  [-model model_name]
  [-domain domain_name]
  [-controlling_domain controlling_domain_name]
  [-transitive transitive]
```

Data Types

<i>element_list</i>	set
<i>exclude_list</i>	set
<i>model_name</i>	string
<i>domain_name</i>	string
<i>controlling_domain_name</i>	string
<i>transitive</i>	Boolean

ARGUMENTS

Synthesis tool does not evaluate the arguments.

DESCRIPTION

This is a simulation command. It is ignored as it does not have functional impact in the synthesis tool. **save_upf** will not write out this command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

SEE ALSO

simplify_constants

Propagates constants and other information in the current design.

SYNTAX

status **simplify_constants**
[-boundary_optimization]

ARGUMENTS

-boundary_optimization

Indicates that **simplify_constants** is to propagate information across all hierarchical boundaries in the design. Use the **set_boundary_optimization** command to perform boundary optimization on a particular set of subdesigns.

DESCRIPTION

This command propagates constants and unconnected information inside each module of the current design and simplifies the logic. When **-boundary_optimization** is used, this information is propagated across all boundaries.

EXAMPLES

In the following example, using library A, the constants are propagated in the current design:

```
prompt> link_library = {A}  
prompt> target_library = {A}  
prompt> read_file -format verilog my_design.v  
prompt> current_design  
prompt> simplify_constants
```

SEE ALSO

compile(2)

```
set_boundary_optimization(2)
set_logic_one(2)
set_logic_zero(2)
set_unconnected(2)
```

size_cell

Relinks leaf cells to a new library cell that has the required drive strength (or other properties).

SYNTAX

```
collection size_cell  
  cell_object  
  lib_cell_object
```

Data Types

```
cell_object    list  
lib_cell_object string
```

ARGUMENTS

cell_object

Specifies the leaf cell to be relinked. Each cell must be in scope; that is, the cell must be at or below the current instance.

lib_cell_object

Specifies the library cell objects to which the specified cell is to be linked. In this case, each object is either a named library cell or a library cell collection. The library specified cells must be logical equivalent to the library cell that is to be relinked.

DESCRIPTION

The **size_cell** command changes the drive strength (or other properties) of a leaf cell by relinking it to a new library cell that has the required properties. Like all other netlist editing commands, all of the arguments of the **size_cell** command must succeed for a successful command run. If any argument fails, the netlist remains unchanged. The **size_cell** command returns the collection of cells sized if successful and a NULL string if unsuccessful. Each cell in *cell_list* must be in scope; that is, the cell must be at or below the current instance.

The *lib_cell* that is being swapped in must conform to the following restrictions:

- *lib_cell* must be functionally compatible with the current library cell to which the cells in *cell_list* are linked.
- *lib_cell* cannot be the same as the current library cell of any of the cells in *cell_list*.
- *lib_cell* must have the same pin count and pin directions as the current library cell of the cells in *cell_list*. The **size_cell** command matches *lib_cell* by its pin numbers.

You can get a list of library cells that are compatible with a given cell using the **get_alternative_lib_cells** command.

The **size_cell** command is for leaf cells only. The **size_cell** command is optimized for incremental timing and does not do a full timing update.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, an attempt to size a cell fails because the target is not functionally equivalent. After finding a compatible library cell, the second attempt succeeds.

```
prompt> size_cell o_reg1 class/NR4P
Error: Could not size 'o_reg1' ('FD2') with 'NR4P'. (NLE-009)
```

```
prompt> get_alternative_lib_cells o_reg1
{class/FD2P}
```

```
prompt> size_cell o_reg1 class/FD2P
Information: Sizing cell 'o_reg1' with lib cell 'class/FD2P'
{o_req1}
```

SEE ALSO

```
get_alternative_lib_cells(2)
get_libs(2)
get_lib_cells(2)
```

sizeof_collection

Returns the number of objects in a collection.

SYNTAX

```
int sizeof_collection  
  [-categorize]  
  collection1
```

Data Types

collection1 collection

ARGUMENTS

-categorize

Return 0, 1, 2 indicating if the collection has 1, or or more than 1 item.

collection1

Specifies the collection for which to get the number of objects. If the empty collection (empty string) is used for the *collection1* argument, the command returns 0.

DESCRIPTION

The **sizeof_collection** command is an efficient mechanism for determining the number of objects in a collection.

EXAMPLES

The following example from PrimeTime shows a simple way to find out how many objects matched a particular pattern and filter in the **get_cells** command.

```
prompt> echo "Number of hierarchical cells: \  
?      [sizeof_collection \  
?      [filter_collection [get_cells * -hier \  
?      "is_hierarchical == true"]]]"  
Number of hierarchical cells is: 10
```

The following example from PrimeTime shows what happens when the argument for the **sizeof_collection** command results in an empty collection.

```
prompt> set s1 [get_cells *]
{u1 u2 u3}
prompt> set ssize [filter_collection $s1 "area < 0"]
prompt> echo "Cells with area < 0: [sizeof_collection $ssize]"
Cells with area < 0: 0
prompt> unset s1
```

SEE ALSO

[collections\(2\)](#)
[filter_collection\(2\)](#)

sort_collection

Sorts a collection based on one or more attributes, resulting in a new, sorted collection. The sort is ascending by default.

SYNTAX

```
collection sort_collection
  [-descending]
  [-dictionary]
  [-limit value_count]
  collection
  criteria
```

Data Types

```
value_count  int
collection   string
criteria     list
```

ARGUMENTS

-descending

Indicates that the collection is to be sorted in reverse order. By default, the sort proceeds in ascending order.

-dictionary

Sort strings dictionary order. For example "a30" would come after "a4".

-limit *value_count*

Only return the first unique values from the primary sort key.

collection

Specifies the collection to be sorted.

criteria

Specifies a list of one or more application or user-defined attributes to use as sort keys.

DESCRIPTION

You can use the **sort_collection** command to order the objects in a collection based on one or more attributes. For example, to get a collection of leaf cells increasing alphabetically, followed by hierarchical cells increasing alphabetically, sort the collection of cells using the *is_hierarchical* and *full_name* attributes as *criteria*.

The *criteria* can specify an attribute on another object. The other object must be available as an attribute on this object. For example, if you had a collection of net shapes, you can sort the objects by net using *owner_net.name* for instance.

In an ascending sort, Boolean attributes are sorted with those objects first that have the attribute set to *false*, followed by the objects that have the attribute set to *true*. In the case of a sparse attribute, objects that have the attribute come first, followed by the objects that do not have the attribute.

Sorts are ascending by default. You can specify ascending or descending order on a per attribute basis. You do this by adding a suffix to the attribute name in the sort criteria. The '-' means sort descending and '+' means sort ascending. If no suffix is specified than the default order ascending unless the *-descending* option is used.

The *-limit* option limits the size of the returned collection by choosing the first unique values from the primary sort criteria. For example using a limit of 1 will return all the objects with the smallest (or biggest if descending) value according to the primary sort key.

EXAMPLES

The following example from PrimeTime sorts a collection of cells based on hierarchy, and adds a second key to list them alphabetically. In this example, cells i1, i2 and i10 are hierarchical, and o1 and o2 are leaf cells. Because the *is_hierarchical* attribute is a Boolean attribute, those objects with the attribute set to *false* are listed first in the sorted collection.

```
prompt> set zc [get_cells {o2 i2 o1 i1 i10}]
{o1 i2 o1 i1 i10}
prompt> set zsort [sort_collection $zc {is_hierarchical full_name}]
{o1 o2i1 i10 i2}
prompt> set zsort [sort_collection -dictionary $zc {is_hierarchical full_name}]
{o1 o2 i1 i2 i10}
prompt> set zsort [sort_collection -dictionary $zc {area- full_name}]
{i10 o1 o2 i1 i2 i10}
prompt> set zsort [sort_collection -dictionary $zc {area- full_name} -limit 1]
{i10 o1}
```

SEE ALSO

[collections\(2\)](#)

source

Read a file and evaluate it as a Tcl script.

SYNTAX

string **source**
[-echo] [-verbose] [-continue_on_error] *file*

Data Types

file string

ARGUMENTS

-echo

Echoes each command as it is executed. Note that this option is a non-standard extension to Tcl.

-verbose

Displays the result of each command executed. Note that error messages are displayed regardless. Also note that this option is a non-standard extension to Tcl.

-continue_on_error

Don't stop script on errors. Similar to setting the shell variable `sh_continue_on_error` to true, but only applies to this particular script.

file

Script file to read.

DESCRIPTION

The **source** command takes the contents of the specified *file* and passes it to the command interpreter as a text script. The result of the source command is the result of the last command executed from the file. If an error occurs in evaluating the contents of the script, then the **source** command returns that error. If a return command is invoked from within the file, the remainder of the file is skipped and the source command returns normally with the result from the return command.

By default, source works quietly, like UNIX. It is possible to get various other intermediate information from the source command using the **-echo** and **-verbose** options. The **-echo** option echoes each command as it appears in the script. The **-verbose** option echoes the result of each command after execution.

NOTE: To emulate the behavior of the `dc_shell` **include** command, use both of these options.

The file name can be a fully expanded file name and can begin with a tilde. Under normal circumstances, the file is searched for based only on what you typed. However, if the system variable `sh_source_uses_search_path` is set to "true", the file is searched for based on the path established with the `search_path` variable.

The **source** command supports several file formats. The *file* can be a simple ascii script file, an ascii script file compressed with gzip, or a Tcl bytecode script, created by TclPro Compiler. The **-echo** and **-verbose** options are ignored for gzip formatted files.

EXAMPLES

This example reads in a script of aliases:

```
prompt> source -echo aliases.tcl
alias q quit
alias hv {help -verbose}
alias include {source -echo -verbose}
prompt>
```

SEE ALSO

`search_path(3)`
`sh_source_uses_search_path(3)`
`sh_continue_on_error(3)`

split_register_bank

Splits a multibit register bank into smaller multibit registers or single-bit registers.

SYNTAX

```
status split_register_bank  
  bank_name  
  -lib_cells list_of_lib_cells
```

Data Types

<i>bank_name</i>	string
<i>list_of_lib_cells</i>	list

ARGUMENTS

bank_name

Specifies the name of the register bank that needs to be split, as specified in the **create_register_bank** command that created the register bank.

-lib_cells *list_of_lib_cells*

Specifies the list of smaller multibit registers or single-bit registers used to replace the multibit register bank, in sequential order of usage, repeated as needed to replace the full number of bits in the bank. Specifying the library name with the cell name (for example, mylib/mreg1) is mandatory in the Design Compiler tool and optional in the IC Compiler tool.

DESCRIPTION

The **split_register_bank** command splits a multibit register bank into smaller multibit registers or single-bit registers. The original register bank is removed from the netlist and replaced by smaller multibit registers or single-bit registers.

The order of the library cells specified by using the **-lib_cells** option determines the pin connection order of the split banks.

The total bit-width of all the listed split banks should equal the bit-width of the original bank. If the total bit-width is too large, some registers of the split banks are not used, and the pins of the unused registers are left dangling.

If a pin in the original bank does not have a corresponding pin in the split bank, the tool disconnects the pin and issues a warning message.

In the Design Compiler tool, you can use the **create_register_bank** and **split_register_bank** commands only before scan insertion. If you want to use these commands after scan insertion, use them later in the IC Compiler tool.

Multicorner-Multimode Support

In designs with multiple scenarios, to properly transfer scenario-specific information during splitting operations, you must activate all scenarios before you run the **split_register_bank** commands.

EXAMPLES

The following example splits a 6-bit register bank into two smaller 3-bit register banks:

```
prompt> split_register_bank reg_6_bit_inst \  
-lib_cells { mylib/REG_3_BIT mylib/REG_3_BIT }
```

SEE ALSO

`create_register_bank(2)`

start_icc2

Launches an IC Compiler II floorplanning session from within Design Compiler Graphical or DC Explorer physical mode.

SYNTAX

```
int start_icc2  
  -f file_name  
  [-check_only]  
  [-verbose]
```

Data Types

file_name string

ARGUMENTS

-f *file_name*

Allows you to run a batch script. This optional argument allows batch processing of a specified file. Make sure that the batch script has a quit command specified at the end of the file or it will wait for user input in the `icc2_shell`.

-check_only

Performs the startup checks and issues errors, if any.

-verbose

Prints out verbose messages, if any.

DESCRIPTION

The **start_icc2** command launches an IC Compiler II session from within Design Compiler Graphical or DC Explorer physical mode, allowing you to create or modify floorplans using the IC Compiler II tool.

Use the **set_icc2_options** command to set the IC Compiler II specific options, such as the executable and the IC Compiler II reference libraries. Use the **report_icc2_options** command to report the **set_icc2_options** command settings. Then, use the **start_icc2** command to launch the IC Compiler II session.

Note: Do not use the **start_icc2** and **start_icc_dp** commands in the same Design Compiler session.

EXAMPLES

In the following example, the **start_icc2** command launches the IC Compiler II session using settings specified by the **set_icc2_options** command:

```
prompt> set_icc2_options -ref_libs "lib1.ndm lib2.ndm" -icc2_executable  
/my_executable_location/icc2_shell  
1  
prompt> start_icc2
```

The following example performs startup checks and issues an error due to missing IC Compiler II settings:

```
prompt> start_icc2 -check_only  
Error: Missing set_icc2_options information. (DCT-158)  
0
```

SEE ALSO

[compile_ultra\(2\)](#)
[set_icc2_options\(2\)](#)
[report_icc2_options\(2\)](#)

start_icc_dp

Launches the floorplan exploration session from Design Compiler Graphical.

SYNTAX

```
int start_icc_dp  
-f file_name  
-check_only flag  
-verbose flag
```

Data Types

```
file_name  string  
flag       flag
```

ARGUMENTS

-f *file_name*

Allows batch processing.

This optional argument allows batch processing of a file specified. Make sure that the batch file has a quit command specified at the end of the file or it will wait for user input in `icc_shell`.

-check_only *flag*

Performs the start up checks and issues errors, if any.

-verbose *flag*

Prints out verbose messages, if any.

DESCRIPTION

This command launches the floorplan exploration session from Design Compiler Graphical. Use the **set_icc_dp_options** command to set the design planning options, and use the **report_icc_dp_options** command to report the **set_icc_dp_options** settings. Then, use **start_icc_dp** to begin the floorplan exploration session.

EXAMPLES

```
prompt> start_icc_dp
```

```
prompt> start_icc_dp -check_only
```

SEE ALSO

```
compile_ultra(2)  
set_icc_dp_options(2)  
report_icc_dp_options(2)
```

streaming_dft_planner

Provides visualization of the current DFTMAX Ultra architecture.

SYNTAX

```
status streaming_dft_planner  
[-show flow | elements | all]  
[-preview_output file_name]
```

DESCRIPTION

This command provides a visualization of the currently configured DFTMAX Ultra architecture. The visualization is based on the current DFT (scan and on-chip clocking (OCC) controller) configuration.

You can modify the DFT configuration and rerun this command as many times as needed until you are satisfied with the architecture.

For more information on the data provided in the planning report, see SolvNet article 2150838, "Understanding The Streaming DFT Planner Report."

This command does not perform any synthesis or optimization tasks.

ARGUMENTS

-show flow | elements | all

Generates the specified report types.

You must specify one of the following values:

- The **flow** argument displays a report that focuses on scan chain lengths and compression ratios. This view summarizes the DFT architecture. It is useful for length-balancing multiple codecs.

This is the default.

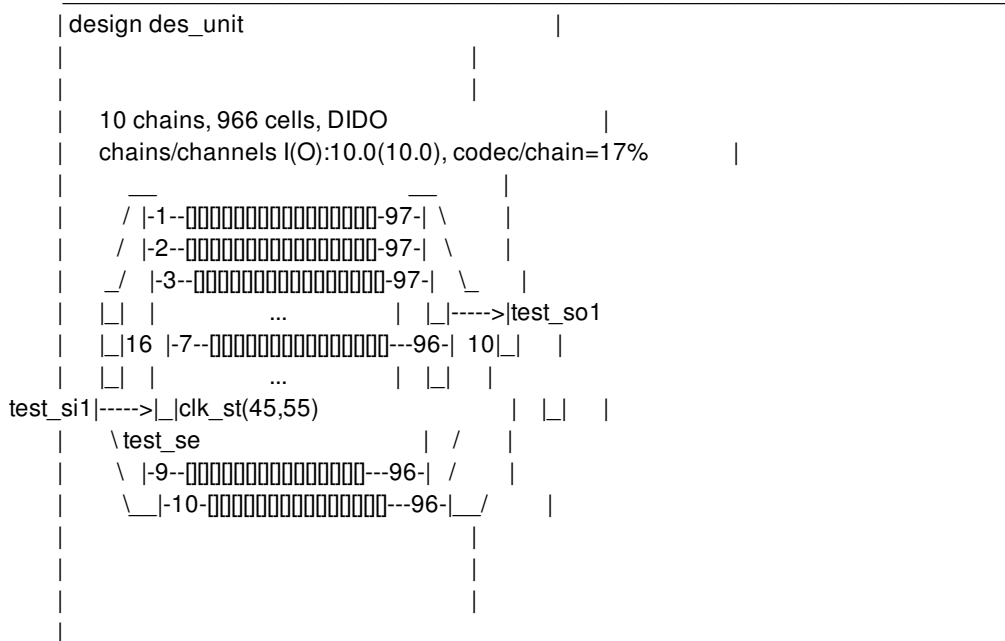
- The **elements** argument displays a report that shows more detail about the elements within the scan chains. It shows information such as clock and polarity information, lock-up latches, retiming registers, and test clock waveform information. This view shows how the clock edges interact in the compression architecture, and it shows why lock-up latches and retiming registers are inserted.
- The **all** argument displays both the **flow** and **elements** reports.

-preview_output *file_name*

EXAMPLES

```
prompt> streaming_dft_planner
      or
prompt> streaming_dft_planner -show flow
```

Compression DFT Flow for test mode: ScanCompression_mode



```

DFT Flow:          TDI
Base scan mode:      Internal_scan
Base scan mode chains: 1
Base scan mode maximum shift length: 966

```

streaming dft planner

The following example displays an **elements** visualization for a TDI flow design:

prompt> **streaming_dft_planner -show elements**

Running Streaming Compression DFT Planner (version 1.0)

Compression DFT Elements for test mode: ScanCompression_mode

Rising edge :r Falling edge :f Lockup latch :L

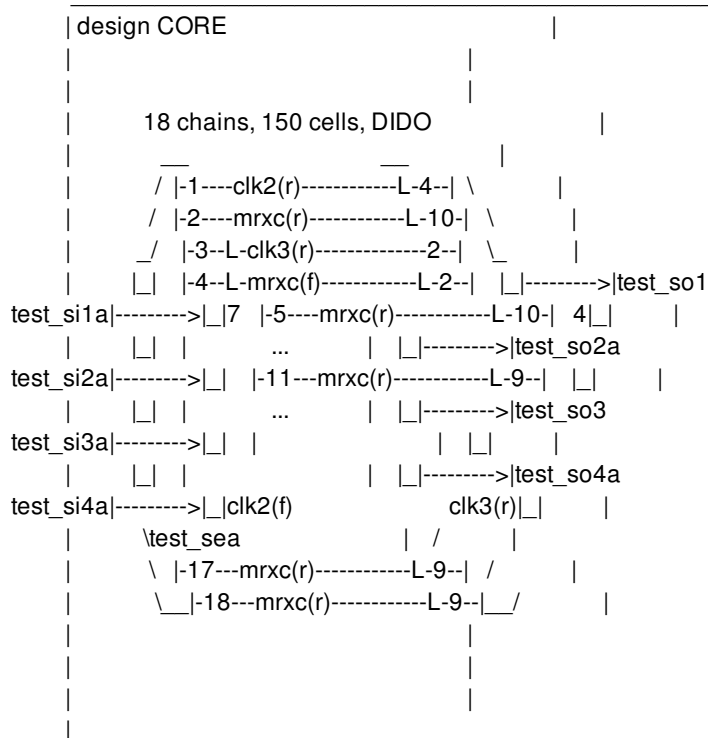
Clocks : clk2, clk3, mrxc

Timing :

clk2(45,55) : _____| | _____

clk3(55,65) : _____| | _____

mrxc(45,55) : _____| | _____



Streaming Compression DFT Elements Information

DFT Flow: TDI

Base scan mode: Internal_scan

Clock with maximum number of flops: mrxc (r, 145 FFs)

Clock with minimum number of flops: mrxc (f, 2 FFs), clk3 (r, 2 FFs)
Scan chain with maximum number of lockups: 4 (2 Ls)
Total number of lockup latches: 19

=====

prompt> **streaming_dft_planner -preview_output out.txt**

or

prompt> **streaming_dft_planner -show all -preview_output /tmp/out.txt**

SEE ALSO

set_streaming_compression_configuration(2)
preview_dft(2)
insert_dft(2)
set_dft_configuration(2)
set_dft_signal(2)
define_test_mode(2)

sub_designs_of

Gets the subdesigns according to the options.

SYNTAX

```
collection sub_designs_of  
  [-hierarchy]  
  [-in_partition | -partition_only]  
  [-dt_only | -ndt_only]  
  [-multiple_instances | -single_instances]  
  [-names_only]  
  design
```

ARGUMENTS

-hierarchy

Returns all subdesigns in the hierarchy of the specified design. By default, only the children of the specified design are returned.

-in_partition

Returns only subdesigns that are part of the partition containing the specified design.

This option and the **-partition_only** option are mutually exclusive.

-partition_only

Looks only at subdesigns that are compile partitions (subdesigns that have the **in_partition** attribute set). Specifically, it returns the direct child compile partitions of the specified design (even if they are more than one level away). With the **-hierarchy** option, it returns all compile partitions in the hierarchy below the specified design.

This option and the **-in_partition** option are mutually exclusive.

-dt_only

Returns only subdesigns that are at least instantiated once with the **dont_touch** attribute set.

This option and the **-ndt_only** option are mutually exclusive.

-ndt_only

Returns only subdesigns that are at least instantiated once with the **dont_touch** attribute not set.

This option and the **-dt_only** option are mutually exclusive.

-multiple_instances

Returns only subdesigns that are instantiated multiple times.

This option and the **-single_instances** option are mutually exclusive.

-single_instances

Returns only subdesigns that are instantiated once.

This option and the **-multiple_instances** option are mutually exclusive.

-names_only

Returns a list of design names, instead of a collection.

design

Specifies the design whose subdesigns will be returned.

DESCRIPTION

The **sub_designs_of** command returns a collection of subdesigns of the specified design (or a Tcl list of design names if the **-names_only** option is specified). By default, the command returns the hierarchical children of the specified design. You can return all subdesigns in the hierarchy by specifying the **-hierarchy** option. In addition, you can filter the results in the following ways:

- subdesigns that are the top-level of a partition (**-partition_only**)
- subdesigns within the partition that contains the specified design (**-in_partition**)
- subdesigns that are at least instantiated once with the **dont_touch** attribute set (**-dt_only**)
- subdesigns that are instantiated at least once with the **dont_touch** attribute not set (**-ndt_only**)
- subdesigns that have multiple instances (**-multiple_instances**)
- subdesigns that have a single instance (**-single_instances**)

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

get_attribute(2)
report_partitions(2)
set_attribute(2)

set_compile_partitions(2)
set_dont_touch(2)
sub_instances_of(2)

sub_instances_of

Gets the subinstances according to the options.

SYNTAX

```
collection sub_instances_of  
  [-hierarchy]  
  [-in_partition] [-partition_only]  
  [-dt_only] [-ndt_only]  
  [-of_references reference_list]  
  [-master_instance]  
  [-names_only]  
  design
```

ARGUMENTS

-hierarchy

Returns all subinstances in the hierarchy of the specified design.

By default, only the children of the specified design are returned.

-in_partition

Returns only subinstances that are part of the partition that contains the specified design.

This option and the **-partition_only** option are mutually exclusive.

-partition_only

Looks only at subinstances that are the top-level designs of compile partitions (subdesigns that have the **in_partition** attribute set). Specifically, it returns the instances that reference the next direct child compile partitions below the specified design (even if the instances are more than one level of hierarchy away). With the **-hierarchy** option, it returns all compile partition instances in the hierarchy below the specified design.

This option and the **-in_partition** option are mutually exclusive.

-dt_only

Returns only subinstances that have the **dont_touch** attribute set.

This option and the **-ndt_only** option are mutually exclusive.

-ndt_only

Returns only subinstances that do not have the **dont_touch** attribute set.

This option and the **-dt_only** option are mutually exclusive.

-of_references *reference_list*

Specifies the reference designs whose instances will be returned.

By default, the **sub_instances_of** command returns instances for all reference designs.

-master_instance

Returns only the master instance when a design has multiple instances.

By default, the **sub_instances_of** command returns all instances of a multiply instantiated design.

-names_only

Returns a list of instance names, instead of a collection.

design

Specifies the design whose subinstances will be returned.

DESCRIPTION

This command returns a collection of instances of the specified design (or a Tcl list of instance names, if the **-names_only** option is specified). By default, it returns only the direct children of the design, if the **-hierarchy** option is specified, all instances of the entire design subtree are returned. If the **-master_only** option is specified, it will return only one instance for multiply instantiated designs. It will pick the master instance (i.e. the one that has the **MasterInstance** attribute set). If none of the qualifying instances (according to the other options) is the master instance, any instance of that design will be returned.

In addition, you can filter the results in the following ways:

- subinstances that are the top-level of a partition (**-partition_only**)
- subinstances within the partition that contains the specified design (**-in_partition**)
- subinstances that have the **dont_touch** attribute set (**-dt_only**)
- subinstances that do not have the **dont_touch** attribute set (**-ndt_only**)
- instances of certain designs (**-of_references**)

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

get_attribute(2)
MasterInstance(3)
report_partitions(2)
set_attribute(2)
set_compile_partitions(2)
set_dont_touch(2)
sub_designs_of(2)

suppress_message

Disables printing of one or more informational or warning messages.

SYNTAX

```
string suppress_message  
    [message_list]
```

Data Types

message_list list

ARGUMENTS

message_list

A list of messages to suppress.

DESCRIPTION

The **suppress_message** command provides a mechanism to disable the printing of messages. You can suppress only informational and warning messages. The result of **suppress_message** is always the empty string.

A given message can be suppressed more than once. So, a message must be unsuppressed (using **unsuppress_message**) as many times as it was suppressed in order for it to be enabled. The **print_suppressed_messages** command displays the currently suppressed messages.

EXAMPLES

When the argument to the **unalias** command does not match any existing aliases, the CMD-029 warning message displays. This example shows how to suppress the CMD-029 message:

```
prompt> unalias q*  
Warning: no aliases matched 'q*' (CMD-029)  
prompt> suppress_message CMD-029  
prompt> unalias q*  
prompt>
```

SEE ALSO

`print_suppressed_messages(2)`
`unsuppress_message(2)`
`get_message_ids(2)`
`set_message_info(2)`

translate

Translates a design from one technology to another.

SYNTAX

```
int translate  
    [-preserve_structure]
```

SYNTAX

```
int translate  
    [-preserve_structure]
```

ARGUMENTS

-preserve_structure

Specifies that the structure of a design is to be preserved during **translate**. The default is false.

DESCRIPTION

The **translate** command translates all components of the current design to components found in the target library. If the current design is hierarchical, all subdesigns below the current design are translated. With this command, designs are converted from one technology to another while preserving as much of the original gate structure as possible. Each cell in the design is replaced with a functionally-equivalent cell from the target library. If no corresponding cell is found, the cell is converted to a Synopsys generic logic form. The replacement cell selection is influenced by preferring or disabling specific library cells using the **set_prefer** and **set_dont_use** commands, and by specifying the types of registers using the **set_register_type** command.

The target libraries are specified by the **target_library** variable. The **local_link_library** of the top-level design is set to **target_library** after the design is linked.

The **translate** command does not operate on cells or designs that have the **dont_touch** attribute. After the translation process, cells that are not successfully translated are reported.

EXAMPLES

In the following example, a design is translated from library *A* to library *B*.

```
prompt> link_library = {A}
```

```
prompt> read -f edif test.edif
```

```
prompt> target_library = {B}
```

```
prompt> translate
```

SEE ALSO

- compile(2)
- current_design(2)
- set_dont_touch(2)
- set_dont_use(2)
- set_prefer(2)
- set_register_type(2)
- set_local_link_library(2)
- uniquify(2)
- target_library(3)

unalias

Removes one or more aliases.

SYNTAX

```
string unalias  
  patterns
```

ARGUMENTS

patterns

Specifies the patterns to be matched. This argument can contain more than one pattern. Each pattern can be the name of a specific alias to be removed or a pattern containing the wildcard characters * and %, which match one or more aliases to be removed.

DESCRIPTION

The **unalias** command removes aliases created by the **alias** command.

EXAMPLES

The following command removes all aliases.

```
prompt> unalias *
```

The following command removes all aliases beginning with f, and the alias rt100.

```
prompt> unalias f* rt100
```

SEE ALSO

[alias\(2\)](#)

ungroup

Removes a level of hierarchy.

SYNTAX

```
status ungroup  
  cell_list | -all  
  [-prefix prefix_name]  
  [-flatten]  
  [-simple_names]  
  [-soft]  
  [-small n]  
  [-force]  
  [-start_level n]  
  [-all_instances]
```

Data Types

```
cell_list    list  
prefix_name string  
n            integer
```

ARGUMENTS

cell_list

Specifies a list of cells in the current design that are to be ungrouped. The contents of these cells are brought up to the same level as the cells. You must specify either *cell_list* or **-all** but not both.

-all

Indicates that all cells in the current design or current instance are to be ungrouped. You must specify either **-all** or *cell_list* but not both.

-prefix *prefix_name*

Specifies the prefix to use in naming ungrouped cells. The default naming style is as follows:

```
cell_being_ungrouped/old_cell_name{number}
```

To provide a vestige of the former hierarchy, omit this option when using the **-flatten** option.

-flatten

Indicates that the specified cell and all of its subcells are to be exploded recursively until all levels of hierarchy are removed.

-simple_names

Indicates that simple, nonhierarchical names are to be used for cells that are ungrouped. Unless this option is used, cells are given default hierarchical names. With this option, cells maintain their original names.

-soft

Indicates that the **group_name** attribute is to be removed on the specified cells. With the **-soft** and **-flatten** options, the **group_name** attribute is removed recursively down the hierarchy of the selected cells. The **group_name** attribute specifies cell grouping constraints for layout placement tools.

-small *n*

Causes all subdesigns with less than the number of leaf cells specified by *n* to be ungrouped. This option implies the **-all** option.

-force

Causes all **dont_touch** hierarchical cells to be flattened. The **dont_touch** attribute is inherited by their leaf cells. This option can only be used when either **-all** or *cell_list* is specified.

-start_level *n*

Causes all hierarchical cells that are at the level specified by *n* to be flattened. This option implies the **-flatten** option. The level value can be 1, 2, 3, and so on. Specifying a value of 1 for **-start_level** flattens the cells from the current design. Specifying a value of 2 causes the top-level blocks to be maintained and all cells in each of the top-level blocks to be flattened. The current instance cannot be set when this option is used. This option cannot be used with the **-small** option.

-all_instances

Enables the command to accept instance cells that are not unique in the object list.

DESCRIPTION

Removes a single level of hierarchy from the current design by exploding the contents of the specified cell or cell instance in the current design.

Any cells that are marked **dont_touch** are left undisturbed. Black boxes are also not collapsed. Clock gates inserted by Power Compiler are not collapsed by default. To do that, set the **power_cg_flatten** variable to TRUE before running **ungroup**.

New cell names are created by concatenating *prefix_name* with original cell names. If the resulting name is not unique, the new unique name is similar to the following:

<prefix_name>cell<number>

If the design contains instances (leaf cells) and only part of the design hierarchy needs to be ungrouped, use **current_instance** to set the instance before ungrouping the cells within that design instance. Alternatively, you can use the full hierarchical name of the cell instance. When **current_instance** is defined, it must be unique and when cell instance is used, the parent must be unique. This restriction prevents design inconsistency.

If you do not specify a prefix, the default is as follows:

cell_being_ungrouped/old_cell_name{number}

If there are dangling nets in the design, they are removed by **ungroup**.

If timing derate is defined for the cell being ungrouped, the derates are pushed down to the lower-level cells if they or their library

cells do not have the corresponding derates. This can cause timing inconsistency in the timing report before and after the ungrouping.

Use the **-all_instances** option when any cell in the object list is an instance in the lower hierarchy and its parent cell is not unique. All similar instance cells under the same parent design are ungrouped. The current design does not have to be changed in order to ungroup the instance cells in the lower design hierarchy. If none of the cells in the object list is an instance in the lower hierarchy, or if its parent cell is unique, the **-all_instances** option is not required.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information. However, in order to preserve constraints and attributes for inactive scenarios, the tool automatically activates inactive scenarios before the hierarchy is changed and deactivates them before the command returns. Due to the overhead, if there are many ungroup commands in the script, it is more runtime efficient to activate inactive scenarios once before the **ungroup** command executions and deactivate them after all the **ungroup** command executions finish.

EXAMPLES

The following example ungroups a specified list of cells:

```
prompt> ungroup {u1 u2 u3}
```

The following example ungroups a specific cell and specifies the prefix to be used:

```
prompt> ungroup {u1} -prefix "U1:"
```

The following example completely removes the hierarchy of two cells:

```
prompt> ungroup {u1 u2} -flatten
```

The following example completely collapses the hierarchy of the current design:

```
prompt> ungroup -all -flatten
```

The following example ungroups all cells in the design except u1:

```
prompt> set_dont_touch {u1}
```

```
prompt> ungroup -all
```

The following example hierarchically removes cell grouping constraints destined to drive layout placement tools without modifying the design hierarchy:

```
prompt> ungroup -soft -flatten
```

The following example ungroups all cells under the instance A/B/C:

```
prompt> current_instance A/B
```

```
prompt> ungroup C
```

The following example flattens all cells starting at level 2:

```
prompt> ungroup -start_level 2
```

The following example ungroups the cell instance MID1/BOT1/FOO1 and MID1/BOT1/FOO2. MID1/BOT1 is a unique instantiation of design BOT.

```
prompt> ungroup {MID1/BOT1/FOO1 MID1/BOT1/FOO2}
```

SEE ALSO

- current_design(2)
- current_instance(2)
- group(2)
- remove_attribute(2)
- set_dont_touch(2)
- power_cg_flatten(3)
- ungroup_keep_original_design(3)
- set_active_scenarios(2)

uniquify

Removes the multiply instantiated hierarchy in the current design by creating a unique design for each cell instance.

SYNTAX

```
status uniquify  
  [-force]  
  [-base_name base_name]  
  [-cell cell_list]  
  [-reference design_name]  
  [-new_name new_design_name]  
  [-dont_skip_empty_designs]
```

Data Types

```
base_name      string  
cell_list      list  
design_name     string  
new_design_name string
```

ARGUMENTS

-force

Renames instances even if they are already unique or are assigned the **dont_touch** attribute. The top-level design and the ILMs are not renamed.

-base_name *base_name*

Specifies the base name to be used instead of the original design name for new designs.

-cell *cell_list*

Specifies a list of cells for which unique designs are to be generated. Unique designs are generated even if the referenced design is already unique, or if the **dont_touch** attribute is set on the cell or its reference. The cells specified in *cell_list* must be in the current design.

-reference *design_name*

Specifies the name of a design that is referenced by cells for which unique designs are to be created. Unique designs are created even if the referenced design is already unique, or if the **dont_touch** attribute is set on the cells or reference. Only cells in the current design that reference the given design are considered.

-new_name *new_design_name*

Specifies the name for a single cell specified in the **-cell** option.

-dont_skip_empty_designs

Creates unique designs even for black box designs. Without this option, the **uniquify** command does not modify black box designs.

DESCRIPTION

The **uniquify** command removes the multiply instantiated hierarchy in the current design by creating a unique design for each cell instance. The command initiates a search for designs referenced in the hierarchy of the current design and generates new, uniquely named designs for all instances that have the same name. If the **dont_touch** attribute is set on a cell, reference, or design, a new design is not created for that cell or any cell with that reference or design.

The **uniquify** command links the current design before it executes. If the link fails, **uniquify** aborts and issues an error message.

If a new design is generated, it is copied from the original design, and placed in the same file as the current design. The new design is named according to the **uniquify_naming_style** variable. The default value of this variable is %s_%d, where %s is replaced by the original design name unless the **-base_name** option is specified, and %d is replaced by the smallest integer that forms a name not used by any other design in memory. The cell reference is updated to use the new design name.

After the **uniquify** command executes, all designs in the hierarchy of the current design have unique names. However, some names might conflict with designs on disk if the current design is a submodule of another design. In this case, the user must resolve these conflicts before integrating the designs. One way to resolve conflicts is to use the **uniquify** command with different values of the **uniquify_naming_style** variable.

When a cell instance is used with the **-cell** option, the complete path to that instance is uniquified (all the parent instances are uniquified if they are not already unique).

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates uniquely named designs for all cells and designs in the hierarchy of the current design that have non-unique names, unless a cell or reference has the **dont_touch** attribute set.

```
prompt> uniquify
```

In the following example, a new design "ADDER1" is created for cell "U4" in the current design.

```
prompt> uniquify -cell U4 -new_name ADDER1
```

In the following example, new designs are created with the base name "X" for all cells starting with "U" in the current design.

```
prompt> uniquify -cell [get_cells U*] -base_name X
```

The following example forces all designs in the hierarchy of the current design to be renamed using the **uniquify_naming_style** variable.

```
prompt> set uniquify_naming_style "joe_%s_%d"
```

```
prompt> uniquify -force
```

The following example uniquifies the instance mid1 and mid1/bot1. Though the argument to uniquify command is mid1/bot1 since the design mid corresponding to mid1 is not unique, mid1 is also uniquified.

```
prompt> uniquify -cell mid1/bot1
```

SEE ALSO

- compile(2)
- current_design(2)
- set_boundary_optimization(2)
- set_dont_touch(2)
- set_dont_use(2)
- uniquify_naming_style(3)
- uniquify_keep_original_design(3)

unset_power_guide

Unsets an existing power guide to be just like an exclusive movebound.

SYNTAX

unset_power_guide
[*power_guide_list*]

Data Types

power_guide_list collection

ARGUMENTS

power_guide_list

Unsets the selected power guides.

DESCRIPTION

This command unsets the given power guides. If no specific name is given, all existing power guides will be unset. They will be in design just like an exclusive movebound.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the flow of the **unset_power_guide** command. In it, if the given power guide AO_WELL exists in the design it will be unset.

```
prompt> unset_power_guide AO_WELL
```

SEE ALSO

`create_bounds(2)`
`create_voltage_area(2)`
`set_power_guide(2)`

unsetenv

Removes a system environment variable.

SYNTAX

```
string unsetenv  
    variable_name
```

Data Types

variable_name string

ARGUMENTS

variable_name

Specifies the name of the environment variable to remove.

DESCRIPTION

The **unsetenv** command searches the system environment for the specified environment variable and removes it from the environment.

If the variable is not defined in the environment, the command returns a Tcl error. The command is catchable.

Environment variables are stored in the **env** Tcl array variable. The **unsetenv** command is a convenience function to interact with this array. It is equivalent to the **unset ::env(*variable_name*)** command.

The application inherits the initial values for the environment variables from its parent process (that is, the shell from which you invoked the application). If you remove the variable using the **unsetenv** command, you remove the variable value in the application and in any new child processes you initiate from the application using the **exec** command. However, the variable is still set in the parent process.

See the man pages for the **set** and **unset** commands for information about working with non-environment variables.

EXAMPLES

In the following example, the **unsetenv** command removes the DISPLAY variable from the environment:

```
prompt> getenv DISPLAY  
host:0  
prompt> unsetenv DISPLAY  
prompt> getenv DISPLAY  
Error: can't read "::env(DISPLAY)": no such variable  
Use error_info for more info. (CMD-013)
```

SEE ALSO

- catch(2)
- exec(2)
- printenv(2)
- set(2)
- unset(2)
- setenv(2)
- getenv(2)

unsuppress_message

Enables printing of one or more suppressed informational or suppressed warning messages.

SYNTAX

```
string unsuppress_message  
    [messages]
```

Data Types

messages list

ARGUMENTS

messages

A list of messages to enable.

DESCRIPTION

The **unsuppress_message** command provides a mechanism to re-enable the printing of messages which have been suppressed using **suppress_message**. You can suppress only informational and warning messages, so the **unsuppress_message** command is only useful for informational and warning messages. The result of **unsuppress_message** is always the empty string.

You can suppress a given message more than once. So, you must unsuppress a message as many times as it was suppressed in order to enable it. The **print_suppressed_messages** command displays currently suppressed messages.

EXAMPLES

When the argument to the **unalias** command does not match any existing aliases, the CMD-029 warning message displays. This example shows how to re-enable the suppressed CMD-029 message. Assume that there are no aliases beginning with 'q'.

```
prompt> unalias q*  
prompt> unsuppress_message CMD-029  
prompt> unalias q*  
Warning: no aliases matched 'q*' (CMD-029)
```

SEE ALSO

`print_suppressed_messages(2)`
`suppress_message(2)`

update_bounds

Updates an existing bound by adding or removing objects.

SYNTAX

```
int update_bounds
  [-name bound_name]
  [-bound bound_object]
  [-add]
  [-remove]
  cell_list
```

ARGUMENTS

-name *bound_name*

Specifies the name of the bound.

-bound *bound_object*

Specifies the bound object. You can specify the bound object directly or by using the **get_bounds** command. You cannot specify more than one bound object.

-add

Adds cells to the bound.

-remove

Removes cells from the bound.

cell_list

Specifies the list of cells to be attached to the bound. If a cell is a hierarchical cell, the bound is also applied to all cells in the subdesigns.

DESCRIPTION

The **update_bounds** command enables you to add cells to or remove cells from an existing move bound or group bound.

For a bound, you can change the floorplan by adding cells within the bound or readjust the utilization by removing cells from the bound. If you add new objects to an existing bound, the placer uses them during coarse placement. When objects are removed from a bound, the placer assumes that the objects are no longer constrained by placement bounds.

When adding objects to a bound, an error occurs if the specified objects are already in a bound. Similarly when removing objects from a bound, an error occurs if the objects do not belong to the same bound.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a bound named `my_bound` with the **`create_bounds`** command. Next, it adds cells to the `my_bound` bound, and then it removes a cell from the `my_bound` bound.

```
prompt> create_bounds -name "my_bound" -coord {100 100 200 200} INSTN_1
prompt> update_bounds -name "my_bound" -add {INSTN_2 INSTN_3}
prompt> update_bounds -bound [get_bounds my_bound] -remove INSTN_3
```

SEE ALSO

- `create_bounds(2)`
- `remove_bounds(2)`
- `report_bounds(2)`
- `get_bounds(2)`

update_cross_probing_files

Updates the paths of the cross-probed files.

SYNTAX

```
status update_cross_probing_files
[-search_path paths]
[-original_file file1]
[-new_file file2]
[-write_script_only output_file]
[-force]
[-verbose]
```

Data Types

```
paths      list
file1      string
file2      string
output_file string
```

ARGUMENTS

-search_path *paths*

By default, the **search_path** variable is used to search for files when the tool updates the cross-probed files. If this option is specified, the tool uses the *paths* specified with this option instead.

When search paths are used, the first matching file found will be used as the replacement file. In general, a file is considered a match if its basename matches the original file and the file content is identical as determined by the tool.

-original_file *file1* -new_file *file2*

Updates only one cross-probed file. The **-original_file** and **-new_file** options must be used together. The existing cross-probed file, *file1*, in the cross-probing database will be updated to the new file specified by *file2*. For the update to be successful,

- (1) The *file1* argument needs to be an existing file in the set of cross-probed files as listed by the **report_cross_probing_files** command.
- (2) The basename of the file in *file1* and *file2* need to match unless the **-force** option is used.
- (3) The calculated checksum value of *file2* needs to match the one stored in the database for *file1*.

-write_script_only *output_file*

Specifies not to update the paths of the cross-probed files but writes the commands to a file. The file can be examined, modified if necessary, and then sourced as a Tcl script.

-force

Updates all cross-probed files (even the file paths are fine) if you specify this option without the **-original_file** and **-new_file** options. Without this option, the command updates files that do not have an OK status as reported by the **report_cross_probing_files** command.

When you specify the **-original_file** and **-new_file** options, the **-force** option allows the command to update the cross-probed file paths even if the basenames are different.

-verbose

Specifies to use verbose mode. The command prints out additional information messages.

DESCRIPTION

In order for GUI cross-probing functionality to work, the files that are stored in the database must be valid. If the design has been moved to a different location or cross-probed files have been moved to a different disk location, the paths that are stored in the database need to be updated accordingly. Running this command updates these paths.

To get a list of all cross-probed files and each file status, run the **report_cross_probing_files** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **-search_path** option with the **update_cross_probing_files** command:

```
prompt> report_cross_probing_files
```

```
*****
Report : cross_probing_files
Design : top
Version: K-2015.06
Date   : Wed Dec 10 14:09:15 2014
*****
```

Status:

OK - No issues found

Missing - File is missing

No Permissions - User has no read permissions

Modified - File contents have been modified

Filename	Status

/common/rtl/bot.v	Missing
/usr/site/joe/designs/mid.v	Missing
/usr/site/joe/designs/top.v	Modified

```
prompt> update_cross_probing_files -search_path /usr/site/common/rtl/
Original /common/rtl/bot.v
```

Replacement /usr/site/common/rtl/bot.v

Original /usr/site/joe/designs/mid.v

Replacement /usr/site/common/rtl/mid.v

Original /usr/site/joe/designs/top.v

Replacement /usr/site/common/rtl/top.v

prompt> **report_cross_probing_files**

Report : cross_probing_files

Design : top

Version: K-2015.06

Date : Wed Dec 10 14:10:15 2014

Status:

OK - No issues found

Missing - File is missing

No Permissions - User has no read permissions

Modified - File contents have been modified

Filename	Status

/usr/site/common/rtl/bot.v	OK
/usr/site/common/rtl/mid.v	OK
/usr/site/common/rtl/top.v	OK

SEE ALSO

report_cross_probing_files(2)

update_floorplan

Updates the floorplan when reading a floorplan script file generated by the **write_floorplan** command.

SYNTAX

int **update_floorplan**

DESCRIPTION

A floorplan script file is created by the **write_floorplan** command. The **update_floorplan** command is included in this script file.

You use the **read_floorplan** command to read the floorplan script file created by the **write_floorplan** command. During this process, any existing floorplan information is removed and replaced by the floorplan information in the script file.

EXAMPLES

The following example writes out the floorplan information to a script file named top.fp.

```
prompt> write_floorplan -all top.fp
```

The following example reads the floorplan information in the top.fp script file and applies it to the design.

```
prompt> read_floorplan top.fp
```

SEE ALSO

write_floorplan(2)
read_floorplan(2)

update_lib

Reads in a specified library file and uses it to update an existing technology, synthetic, or symbol library.

SYNTAX

```
int update_lib  
  [-overwrite] [-permanent] library_name file_name [-no_warnings]
```

Data Types

```
library_name  string  
file_name     string
```

ARGUMENTS

-overwrite

Indicates that a group declared in *file_name* is to overwrite any group in *library_name* that has the same name. Only groups added by **update_lib** or **model** can be overwritten. If **-force** is specified, some library-level groups(wire_load, power_supply and operating_conditions) included in the original library can be modified. The other groups included in the original library and groups added with **-permanent** cannot be modified.

-permanent

Indicates that groups declared in *file_name* are to be stored in the library in such a way that they cannot be overwritten.

library_name

Specifies the name of the library to be updated. The library must be resident in memory.

file_name

Specifies the name of the file in which one or more new groups are listed. The syntax of this file is the same as that expected by **read_lib**, except that the file contains only library level groups without the library() {} group definition.

-no_warnings

Indicates that all messages of severity 'warning' are to be suppressed. The default is to issue all warning messages. Warning messages can identify serious library problems; use this flag sparingly.

DESCRIPTION

Reads in a library file and adds the groups declared in that file to the library specified. Groups are added to the library as if entered at the end of the original text of the library.

Only library-level group statements are added to a library. Library level functions or attributes are not accepted (some library-level attributes can be an exception, as described in the "-force" section).

The library can be a technology, symbol, or synthetic library. Only groups normally found in a library of that type are accepted. For example, you cannot add a symbol group to a technology library. For technology libraries, the delay model and technology associated with the original library must be the same used by new group(s).

If Library Compiler finds any errors while processing a group, it does not add that group to the library. Legal groups are added. The update operation does not fail if there are illegal groups intermixed with legal groups.

If you update a library with a `wire_load` group describing a wire load model in use by the current design, the wire load model of the current design is reset.

library_name can have a simple or a complex file name. A simple file name has no directory specification, which in UNIX means that it does not contain a "/" (slash). Simple filenames such as **test.lib** or **library.db** must be found in a directory listed in the **search_path**.

A complex file name has a directory specification, which in UNIX means that it contains a slash. Relative or complex filenames such as **./test.lib**, or **~synopsys/dc/test.lib** are not included in the **search_path**.

For details on library file syntax, refer to the *Library Compiler Reference Manual*. If you do not have a Library Compiler License, function statements and state information are ignored when you attempt to update technology library files. You do not need a Library Compiler License to update symbol library files.

Do not use **update_lib** on the synthetic library **standard.sldb**, because its parts are licensed differently from user parts.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, the library file **add_ms.lib**, which contains technology information in Synopsys technology library entry format, is added to the library **cmos**:

```
prompt> update_lib cmos add_ms.lib
```

In the following example, the library file **new_wire_load.lib**, which contains a new **wire_load** group for a technology library, is added to the library **cmos**. Any **wire_load**(s) in the library that have identical names are overwritten if they are not permanent groups. In addition, all warning messages are suppressed with the **-no_warnings** flag.

```
prompt> update_lib -overwrite cmos new_wire_load.lib -no_warnings
```

In the following example, the library file **ramgen.lib**, which contains a black box cell description of a RAM cell, is added to the library **cmos**. The **-permanent** flag indicates that the new cell cannot be overwritten.

```
prompt> update_lib -permanent cmos ramgen.lib
```

SEE ALSO

read_lib(2)
write_lib(2)

update_lib_model

Updates the library to conform to the PG pin library syntax. This command gets the PG information either from reference library FRAM view or from specific Tcl commands. The library voltage, PG pin of the cell, and related PG pin of the cell pin are updated.

SYNTAX

```
status update_lib_model  
-reference_mode FRAM | TCL  
[library_name]
```

ARGUMENTS

-reference_mode FRAM | TCL

Sets the reference mode from where the power and ground pin information is read.

If set to **FRAM**, the command gets the power and ground pin information from the FRAM view of each library cell. When you use this mode, either the Milkyway design library must contain the reference library set or the **mw_reference_library** variable must be set to the reference library.

If set to **TCL**, you must define the voltage and PG pin reference by using the following commands: **update_lib_voltage_model**, **update_lib_pg_pin_model**, and **update_lib_pin_model**.

If the library to be updated uses the old PG syntax, the voltage value is used from the power_supply attachment and rail_connection is used to define the related PG pins.

This is a required option.

library_name

Specifies the library to be converted.

If you do not specify this option, all libraries in the link library or target library set are updated.

DESCRIPTION

The **update_lib_model** command updates the specified library to a Liberty PG pin library. The library can be a non-PG library or it can use the old PG syntax. The reference base for the PG pin definition and association can be either the FRAM reference libraries or the settings from the **update_lib_voltage_model**, **update_lib_pg_pin_model**, and **update_lib_pin_model** commands.

EXAMPLES

The following example sets the voltage map for the library named `standard_lib` and updates the library to a Liberty PG pin library.

```
prompt> update_lib_voltage_model standard_lib \  
       -voltage {VDD 1.2} -voltage {VSS 0.0}  
prompt> update_lib_model -reference_mode TCL standard_lib
```

The following example uses a reference library to update all the libraries in the link library and target library set.

```
prompt> update_lib_model -reference_mode FRAM
```

SEE ALSO

- `update_lib_voltage_model(2)`
- `update_lib_pg_pin_model(2)`
- `update_lib_pin_model(2)`
- `mw_reference_library(3)`

update_lib_pg_pin_model

Defines the power and ground pin model for a library cell.

SYNTAX

```
status update_lib_pg_pin_model
  cell_name
  -pg_pin_name pin_names
  [-pg_voltage_name voltage_names]
  [-pg_pin_type pin_types]
  [-pg_pin_direction pin_directions]
  [-pg_physical_connection physical_connections]
  [-pg_related_bias_pin related_bias_pin]
```

Data Types

<i>cell_name</i>	string
<i>pin_names</i>	list
<i>voltage_names</i>	list
<i>pin_types</i>	list
<i>pin_directions</i>	list
<i>physical_connections</i>	list

ARGUMENTS

cell_name

Specifies the name of the library cell for which to update the power and ground pins.

The format of the *cell_name* argument is [*library_name*]/*cell_name*. If you do not specify the library name, the PG pin model applies to all libraries.

-pg_pin_name *pin_names*

Specifies the names of the power and ground pins of the library cell.

This is a required argument.

-pg_voltage_name *voltage_names*

Specifies the voltage name of the corresponding power or ground pin of the library cell. The voltage names are defined in the library voltage map by using the **update_lib_voltage_model** command.

There must be a one-to-one correspondence between the voltage names specified in this option and the pins specified in the **-pg_pin_name** option.

-pg_pin_type *pin_types*

Specifies the pin type for the each pin specified in the **-pg_pin_name** option. Valid values are `primary_power`, `primary_ground`, `backup_power`, `backup_ground`, `internal_power`, `internal_ground`, `pwell`, `nwell`, `deeppwell`, and `deepnwell`.

There must be a one-to-one correspondence between the pin types specified in this option and the pins specified in the **-pg_pin_name** option.

-pg_pin_direction *pin_directions*

Specifies the direction for each pin specified in the **-pg_pin_name** option. Valid values are `input`, `output`, `inout`, and `internal`.

There must be a one-to-one correspondence between the pin directions specified in this option and the pins specified in the **-pg_pin_name** option.

-pg_physical_connection *physical_connections*

Specifies the type of physical connection used for each pin specified in the **-pg_pin_name** option. Valid values are `device_layer` and `routing_pin`.

There must be a one-to-one correspondence between the connection types specified in this option and the pins specified in the **-pg_pin_name** option.

-pg_related_bias_pin *related_bias_pin*

Specifies the related bias pin for each pin specified in the **-pg_pin_name** option.

There must be a one-to-one correspondence between the related bias pins specified in this option and the pins specified in the **-pg_pin_name** option.

DESCRIPTION

The **update_lib_pg_pin_model** command defines the power and ground pin model for a library cell. The model is later used by the **update_lib_model** command. If the power or ground pin already exists, the new specification overrides the old one. Otherwise, a new power or ground pin is created for the library cell.

EXAMPLES

The following example defines the power and ground pin model for the LIB1/CELL1 library cell.

```
prompt> update_lib_pg_pin_model LIB1/CELL1 -pg_pin_name {VDD VSS} \
      -pg_voltage_name {VDDL VSS} -pg_pin_type {primary_power primary_ground}
```

SEE ALSO

`update_lib_voltage_model(2)`
`update_lib_pin_model(2)`
`update_lib_model(2)`

update_lib_pin_model

Sets the pin map for the specified library cell.

SYNTAX

```
status update_lib_pin_model
      cell_name
      -pins list_of_pins
      [-related_power_pin list_of_pins]
      [-related_ground_pin list_of_pins]
      [-related_bias_pin list_of_pins]
      [-power_down_function power_down_func]
```

Data Types

```
cell_name      string
list_of_pins   list
power_down_func string
```

ARGUMENTS

cell_name

Specifies the name of the library cell for which to set the pin map.

The format for specifying the cell name is *library_name/cell_name*. If you omit the library name, the pin map applies to all libraries.

This is a required argument.

-pins *list_of_pins*

Specifies the names of the pins being mapped. You can specify one or more pins.

This is a required argument.

-related_power_pin *list_of_pins*

Specifies the related power pin for each pin specified in the *-pins* option. There must be a one-to-one correlation between the pins specified in this option and those specified the **-pins** option.

-related_ground_pin *list_of_pins*

Specifies the related ground pin for each pin specified in the *-pins* option. There must be a one-to-one correlation between the pins specified in this option and those specified the **-pins** option.

-related_bias_pin *list_of_pins*

Specifies the related bias pin for each pin specified in the *-pins* option. There must be a one-to-one correlation between the pins specified in this option and those specified the **-pins** option.

-power_down_function *power_down_func*

Specifies the power down function for each pin specified in the *-pins* option. There must be a one-to-one correlation between the pins specified in this option and those specified the **-pins** option.

DESCRIPTION

The **update_lib_pin_model** command sets the pin map for the specified library cell. The pin map is used by the **update_lib_model** command.

If a pin map already exists for the specified library cell, the new specifications overrides the old one. Otherwise, a new pin map is created for the cell.

EXAMPLES

The following example sets the pin map for the CELL1 cell in the LIB1 library.

```
prompt> update_lib_pin_model LIB1/CELL1 -pins {A B} \  
-related_power_pin {VDD VDD} -related_ground_pin {VSS VSS}
```

SEE ALSO

update_lib_voltage_model(2)
update_lib_pg_pin_model(2)
update_lib_model(2)

update_lib_voltage_model

Sets the voltage model for the specified library.

SYNTAX

```
status update_lib_voltage_model
      library_name
      [-voltage {voltage_name voltage_value}]
```

Data Types

<i>library_name</i>	string
<i>voltage_name</i>	string
<i>voltage_value</i>	float

ARGUMENTS

library_name

Specifies the name of the library for which to define the voltage model.

-voltage { *voltage_name* *voltage_value* }

Sets the voltage name and value pair for the specified library. You can specify this option multiple times in a single run of the command.

DESCRIPTION

The **update_lib_voltage_model** command sets the voltage map for the specified library. The map is used later during the **update_lib_model** command.

EXAMPLES

The following example defines the voltage model for a single-rail library.

```
prompt> update_lib_voltage_model standard_lib \
-voltage {VDD 1.2} \
-voltage {VSS 0.0}
```

The following example defines the voltage model for a multiple-rail library.

```
prompt> update_lib_voltage_model standard_lib \  
-voltage {VDD 1.2} \  
-voltage {VSS 0.0} \  
-voltage {VDD_BACK 0.7}
```

SEE ALSO

update_lib_model(2)
update_lib_pg_pin_model(2)
update_lib_pin_model(2)

update_timing

Updates timing information on the current design.

SYNTAX

status **update_timing**

ARGUMENTS

This command has no arguments.

DESCRIPTION

Updates timing for the current design. Timing information is obsoleted when environmental information, such as arrival times or loads, is changed. The **update_timing** command is used to update this information after these changes are made.

Timing information is automatically updated in the current design when **compile** or **translate** is used or when reports that display timing information are used. So, for most cases, the **update_timing** command is not needed. However, there are instances when timing might not be current (such as if **set_attribute** is incorrectly used instead of **set_drive** to set a drive value).

Note that this command does not automatically identify and define clocks in the design. These should be defined by using the **create_clock** command.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

This example updates timing for the current design.

```
prompt> update_timing
```

SEE ALSO

compile(2)
create_clock(2)
set_drive(2)
translate(2)

upf_version

Accepts and preserves a UPF version string.

SYNTAX

```
string upf_version  
[version]
```

Data Types

<i>version</i>	String

ARGUMENTS

version

The UPF version string, such as "2.1". The tool accepts the string without checking.

DESCRIPTION

This command accepts and preserves a string that describes the UPF version number. The command always returns the string "IEEE-1801".

This command does not have any effect aside from preserving the entered string. When you use the **save_upf** command, the command is written to the UPF file exactly as originally entered.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following **upf_version** command saves the UPF version string "2.1".

```
prompt> upf_version 2.1  
IEEE-1801  
...  
prompt> save_upf out.upf
```

1

The UPF file "out.upf" written by the **save_upf** command contains the command "upf_version 2.1".

SEE ALSO

save_upf(2)

use_interface_cell

Specifies how to map the isolation, level-shifter, and enable level-shifter cells belonging to the specified isolation and/or level-shifter strategy.

SYNTAX

```
string use_interface_cell  
  interface_implementation_name  
  -strategy list_of_isolation_level_shifter_strategies  
  -domain power_domain  
  -lib_cells lib_cells  
  [-force_function]  
  [-port_map port_map_list]
```

Data Types

<i>interface_implementation_name</i>	string
<i>list_of_isolation_level_shifter_strategies</i>	list
<i>power_domain</i>	string
<i>lib_cells</i>	list

ARGUMENTS

interface_implementation_name

Specify the name of the interface cell implementation strategy.

-domain *power_domain*

Specifies the name of the power domain name to which the strategy/strategies belongs.

-lib_cells *lib_cells*

Specifies a list of the target library cells to be used for the isolation mapping.

DESCRIPTION

This command defines how the **compile** command performs the mapping of isolation, level-shifter, or enable level-shifter cells.

EXAMPLES

In the following example, the isolation cells belonging to the strategy named ISO1 will be mapped to the isocell1 library cell with the highest priority, and level-shifter cells belonging to the strategy LS1 will be mapped to the levshicell1 library cell with the highest priority:

```
prompt> use_interface_cell my_interface -domain PD1 -strategy {ISO1 LS1} \  
-lib_cells {isocell1 levshicell1} \  
-port_map {{VDD sn1} {VSS gnd}}
```

SEE ALSO

set_isolation(2)
set_level_shifter(2)

use_test_model

Specifies the subdesigns that will use test model information during DRC and DFT insertion.

SYNTAX

```
status use_test_model  
  [-true design_list]  
  [-false design_list]
```

ARGUMENTS

-true *design_list*

Specifies the set of designs for which test models are to be used. This is the default.

-false *design_list*

Specifies the set of designs for which test models are not to be used.

DESCRIPTION

This command is used to specify whether or not DFT Compiler uses Core Test Language (CTL) test model information during DRC and DFT insertion for the specified subdesigns.

The default is to use test models for all designs where test model information is available. When a previously DFT-inserted subdesign contains test model information, such as when it is loaded from a .ddc file, DFT Compiler automatically uses the stored test model information for subsequent DFT operations. The use of test models improves the capacity and runtime of the tool.

When test models are disabled for a subdesign, DFT Compiler ignores the test model information describing complex test logic. However, any test-ready attributes on scan flip-flops are still considered. If the design contains only standard scan chains, DFT Compiler will restitch the chains according to the top-level scan requirements.

You should use caution when disabling test models for a design containing complex test logic, such as core wrapping, on-chip clocking (OCC) controllers, or any type of scan compression. When test models are disabled, these complex structures are no longer meaningful to DFT Compiler, and are simply treated as part of the existing design logic.

EXAMPLES

The following example specifies that test models should not be used for the des3 and des4 designs. Test models will be used for all other designs in memory.

```
prompt> use_test_model -false {des3 des4}
```

SEE ALSO

- current_design(2)
- dft_drc(2)
- insert_dft(2)
- preview_dft(2)
- read_test_model(2)
- reset_design(2)

which

Locates a file and displays its pathname.

SYNTAX

string **which**
filename_list

Data Types

filename_list list

ARGUMENTS

filename_list

List of files to locate.

DESCRIPTION

Displays the location of the specified files. This command uses the `search_path` to find the location of the files. This command can be a useful prelude to `read_db` or `link_design`, because it shows how these commands expand filenames. The **which** command can be used to verify that a file exists in the system.

If an absolute pathname is given, the command searches for the file in the given path and returns the full pathname of the file.

EXAMPLES

The following examples are based on the following `search_path`.

```
prompt> set search_path "/u/foo /u/foo/test"
```

The following command searches for the file name `foo1` in the `search_path`.

```
prompt> which foo1
/u/foo/foo1
```

The following command searches for files `foo2`, `foo3`.

```
prompt> which {foo2 foo3}  
/u/foo/test/foo2 /u/foo/test/foo3
```

The following command returns the full pathname.

```
prompt> which ~/test/designs/sub_design.db  
/u/foo/test/designs/sub_design.db
```

SEE ALSO

read_db(2)
search_path(3)

while

Loop execution control structure.

SYNTAX

return_val **while**

ARGUMENTS

The **while** command has no arguments.

DESCRIPTION

This command iteratively executes commands based on the value of an evaluated *expression*. The format of the **while** statement is similar to that used in the C programming language.

The *expression* is evaluated upon execution. While the *expression* evaluates to a non-zero value, the *loop-statement-block* executes, then the condition *expression* is evaluated again. When the *expression* evaluates to zero, the loop exits.

The return value of the **while** statement is the return value of the last statement executed. If no statements execute, the return value is 0.

The **break** and **continue** commands modify the standard execution of the **while** statement. See the man pages for the **break** and **continue** commands for more information.

The output of the **while** statement can be redirected to a file. This file contains the output of any of the statements executed in the body of the **while** statement. For more information on file redirection, refer to the Tcl documentation.

EXAMPLES

The following example prints squared values from 0 to 10.

```
set p 0
while {$p <= 10} {
    echo "$p squared is: [expr $p * $p]"; incr p
}
```

SEE ALSO

`break(2)`

`if(2)`

win_select_objects

Creates a collection of objects equivalent to a graphical selection operation.

SYNTAX

```
string win_select_objects  
  [-slct_targets slct_bus]  
  [-slct_targets_operation operation]  
  [-create_slct_buses]  
  [-root instance]  
  [-within rectangle  
  | -line line  
  | -at point  
  | -radius r | -again_at]  
  [-intersect]  
  [-index i]  
  [-visible]
```

Data Types

<i>slct_bus</i>	string
<i>operation</i>	string
<i>instance</i>	string

ARGUMENTS

-slct_targets *slct_bus*

Specifies the name of a selection bus to store the result in. By default the result of this command is returned in a collection. If this option is specified selection bus *slct_bus* is used instead. *slct_bus* is also returned as result of the command.

-slct_targets_operation *operation*

Specifies which operation should be used when storing the result in selection bus *slct_bus*. This is only allowed if you specify the -slct_targets option. Legal operations are clear, add and remove.

-create_slct_buses

Specifies that a new selection bus is created and used to store the result in. Using this option is equivalent to first creating a selection bus using the create_selection_bus command and then providing the created selection bus to option -slct_targets.

-root *instance*

Specifies a string for the instance whose component design objects and hierarchy are to be examined against the specified region criteria.

-within *rectangle*

Collects design objects lying within the specified rectangle.

-line *line*

Collects design objects intersected by specified line.

-at *point*

Collects design objects whose bounding box contains the specified point.

-radius *r*

Specifies a radius for points specified by the **-at** option. It is applicable only for **_pins** and **_ports**.

-again_at

Reapplies the previous **-at** option, but returns a previously-matched design object.

-intersect

Collects design objects intersecting with specified rectangle. Without this option, the command collects design objects contained in the specified rectangle.

-index *i*

For point select (-at option) specifies a specific object index to select. Similar to running **-again_at**.

-visible

Select objects marked as visible with **win_set_select_class** command.

DESCRIPTION

This command is for use by the graphics window(s) for logging interactive, graphical selection operations to the command log for later playback. You are advised not to use this command directly. Use of this command other than by graphics windows may cause unexpected results from selection operations.

SEE ALSO

win_set_select_class(2)
win_set_filter(2)

win_set_filter

Sets a filter to apply to objects selected by the **win_select_objects** command.

SYNTAX

```
int win_set_filter  
  -class class_name  
  [-level level]  
  [-filter expression]  
  [-layer list]  
  [-user_filter true | false]  
  [-user_filter_cmd tcl_cmd]  
  [-highlighted_only true | false]  
  [-expand_cell_types list]  
  [-visible]
```

ARGUMENTS

-class *class_name*

Specifies a single class name for which the filter is to apply.

-level *level*

Specifies the hierarchy level, up to which **win_select_objects** searches for design objects down the hierarchy. This has no effect on objects that are not collectable.

-filter *expression*

Specifies a filter expression that an object must satisfy to be returned by the **win_select_objects** command. The expression argument must be a Boolean expression based on attributes of the specified class. If this option is not specified, the filter is cleared.

-layer *list*

Specifies a layer number list that an object must satisfy to be returned by the **win_select_objects** command. If this option is not specified, the layer filter is cleared.

-user_filter true | false

Enables user supplied filter command.

-user_filter_cmd *tcl_cmd*

Specifies a tcl command for filtering.

-highlighted_only true | false

Specifies that only highlighted objects can be selected.

-expand_cell_types *list*

Specifies cell types for searching design objects down the hierarchy when the search level is greater than 0. This has no effect on types that is not supported or objects that are not collectable.

-visible

Specified filters are for visible objects. By default the filters are for selectable objects.

DESCRIPTION

This command is for use by the graphics window(s) for logging interactive, graphical selection operations to the command log for later playback. You are advised not to use this command directly. Use of this command other than by graphics windows may cause unexpected results from selection operations.

SEE ALSO

win_select_objects(2)
win_set_select_class(2)

win_set_select_class

Sets the design objects to be collected by the **win_select_objects** command.

SYNTAX

```
string win_set_select_class  
    {-all | class_names}  
    [-visible]
```

ARGUMENTS

-all

Selects all classes. The **-all** option and the *class_names* option are mutually exclusive.

class_names

Specifies a list of design objects. The **-all** option and the *class_names* option are mutually exclusive.

-visible

Specified classes are for visible objects. By default the classes are for selectable objects.

DESCRIPTION

This command is for use by the graphics window(s) for logging interactive, graphical selection operations to the command log for later playback. You are advised not to use this command directly. Use of this command other than by graphics windows may cause unexpected results from selection operations.

SEE ALSO

win_select_objects(2)
win_set_filter(2)

write

Alias for the **write_file** command.

DESCRIPTION

The **write** command has been renamed **write_file**. The **write** command is provided so that older scripts will run correctly, but it is recommended that you use the command's actual name, **write_file**.

See the **write_file** man page for information about this command.

SEE ALSO

`write_file(2)`

write_app_var

Writes a script to set the current variable values.

SYNTAX

```
string write_app_var  
-output file  
[-all | -only_changed_vars]  
[pattern]
```

Data Types

```
file      string  
pattern  string
```

ARGUMENTS

-output *file*

Specifies the file to which to write the script.

-all

Writes the default values in addition to the current values of the variables.

-only_changed_vars

Writes only the changed variables. This is the default when no options are specified.

pattern

Writes the variables that match the specified *pattern*. The default is "".

DESCRIPTION

The **write_app_var** command generates a Tcl script to set all application variables to their current values. By default, variables set to their default values are not included in the script. You can force the default values to be included by specifying the **-all** option.

EXAMPLES

The following is an example of the **write_app_var** command:

```
prompt> write_app_var -output sh_settings.tcl sh*
```

SEE ALSO

[get_app_var\(2\)](#)
[report_app_var\(2\)](#)
[set_app_var\(2\)](#)

write_bsd_rtl

Writes the RTL for the configured boundary-scan design into an output file.

SYNTAX

```
status write_bsd_rtl  
  [-output output_file_name]  
  [-format verilog]  
  [-all]  
  [-tap]  
  [-bsr]
```

Data Types

output_file_name string

ARGUMENTS

-output *output_file_name*

Specifies the name of the boundary-scan output RTL file to write. If only a base name is given without a directory path, the files are written in the current directory. This option is required.

-format verilog

Selects the format for the output RTL file. The only allowed value is **verilog**, which is the default.

-all

Specifies that all boundary-scan logic modules are to be written to the specified output file. This is the default.

-tap

Specifies that the TAP controller and its submodules should be written to a separate file. A file with `tap_` prefixed to the specified output file name (for example, `tap_rtl_file.v`) is written for the TAP controller modules. The remaining boundary-scan modules are written to the file specified with the **-output** option.

-bsr

Specifies that the boundary-scan register (BSR) and its submodules should be written to a separate file. A file with `bsr_` prefixed to the specified output file name (for example, `bsr_rtl_file.v`) is written for the BSR modules. The remaining boundary-scan modules are written to the file specified with the **-output** option.

DESCRIPTION

The **write_bsd_rtl** command writes the configured boundary-scan design to an output Verilog RTL file. This file can be used for simulation.

The Verilog RTL file can be simulated along with the patterns, as well as analyzed statically with the **check_bsd** command.

The **-tap** and **-bsr** options control how the modules are written to different output RTL files, but they do not affect the actual logic structure of the boundary-scan design.

EXAMPLES

The following example commands write a boundary-scan configured RTL file for a design. Both commands are equivalent.

```
prompt> write_bsd_rtl -format verilog -all -output rtl_file.v
prompt> write_bsd_rtl -output rtl_file.v
```

The following example writes a boundary-scan configured RTL file for a design. The TAP controller and its submodules are written to the tap_rtl_file.v file. The BSR chain and its submodules are written to the bsr_rtl_file.v file. The remaining modules are written to the rtl_file.v file.

```
prompt> write_bsd_rtl -format verilog \
    -tap -bsr -output rtl_file.v
```

SEE ALSO

- set_bsd_configuration(2)
- current_design(2)
- insert_dft(2)
- create_bsd_patterns(2)
- write_test(2)

write_bsdI

Generates the boundary-scan description language (BSDI) file for a boundary-scan design.

SYNTAX

```
int write_bsdI  
  [-naming_check VHDL | BSDI | none]  
  [-output file_name]  
  [-effort low | medium | high]
```

Data Types

file_name string

ARGUMENTS

-naming_check VHDL | BSDI | none

Specifies the type of naming checks to perform on the design. *VHDL* (the default) checks for both VHDL and BSDI reserved words; *BSDI* checks for BSDI reserved words only. *none* disables all naming checks.

-output *file_name*

Specifies the name of the output BSDI file. The default is *top_level_design_name*.bsdI.

-effort low | medium | high

Controls the effort used to search for implemented instructions if the boundary-scan database has not been previously built by the **check_bsd**, **insert_dft**, or **read_bsdI** command and the tool must infer the implemented instructions. For more information, see the man page for the **check_bsd** command.

DESCRIPTION for all modes

Generates the BSDI file for a boundary-scan design. If the boundary-scan database was not previously built by the **check_bsd**, **insert_dft**, or **read_bsdI** command, the command performs an implicit compliance check before proceeding. If a valid port-to-pin map exists for the boundary-scan design and if no errors occurred during compliance checking, the **write_bsdI** command generates the BSDI output file for the boundary-scan design.

You can change the default suffix name, **bsdI**, by setting the **test_bsdI_default_suffix_name** variable with another suffix name.

By default, the BSDI file line length is 74 characters per line. You can specify a different line length using the **test_bsdI_max_line_length** variable. The specified maximum line length should not be less than 50 or more than 132. If you

specify a line length outside this range, **write_bsd** uses the default instead.

EXAMPLES

The following example generates the BSDL output file test.bsd for a boundary-scan design while performing BSDL naming checks on the identifiers used in the design.

```
prompt> write_bsd -naming_check BSDL -out test.bsd
```

SEE ALSO

- check_bsd(2)
- current_design(2)
- read_pin_map(2)
- remove_pin_map(2)
- set_bsd_compliance(2)
- test_bsd_default_suffix_name(3)
- test_bsd_max_line_length(3)

write_cell_expansion

Writes the cell expansion data, including the area, width, and height of each cell that is expanded.

SYNTAX

```
int write_cell_expansion  
    output_file_name
```

Data Types

output_file_name string

ARGUMENTS

output_file_name

Specifies the name of the output expansion data file. This is a required argument.

DESCRIPTION

The **write_cell_expansion** command writes the cell expansion data to a file. Cell expansion data includes the area, width, and height of each cell that is expanded. The data can be used for better congestion correlation between the Design Compiler Graphical tool and the IC Compiler and IC Compiler II tools.

EXAMPLES

The following example writes out a cell expansion file from the existing design.

```
prompt> write_cell_expansion out.exp
```

SEE ALSO

[read_cell_expansion\(2\)](#)

write_collection

Output a machine readable report of attribute values for elements in a collection.

SYNTAX

write_collection

```
collection  
-file filename]  
[-format format]  
[-max_rows value_count]  
[-columns attribute_list]  
[-metadata]
```

Data Types

<i>collection</i>	string
<i>filename</i>	string
<i>format</i>	string
<i>value_count</i>	int
<i>attribute_list</i>	list

ARGUMENTS

collection

Specifies the collection on which to report. The collection may be homogeneous or heterogeneous. If the collection is very large, you may want to use the **-max_rows** option to limit the size of your output.

-file *filename*

This option indicates the name of the file to which the data is written.

-format *format*

This option accepts one of the following valid argument values: "csv" | "tsv". If the option is omitted, the default format is "csv", or command separated values. The argument "tsv" produces a tab separated values formatted data.

-max_rows *value_count*

This option indicates how many rows of data to include in the output. This number indicates the number of collection elements on which data is output.

-columns *attribute_list*

Indicates the set of attributes for which to output values, and their order in the output.

The special attribute name "object_class" may be used to include the element object class names, such as "cell" or "net", in the output.

If the option is omitted, then the object names and object class (or type) names are output by default.

-metadata

This option indicates that a meta data section should be included in the output.

DESCRIPTION

This command outputs tabular data of attribute values for elements of the given collection. The attribute values in the output are determined by the list of attributes given to the **-columns** option. The columns in the tabular report will be ordered by the order of attributes in the list. The command **list_attributes** may be called to view a list of attributes defined for an object class.

If the collection is large, the output size may be limited by indicating a **-max_rows** value. The commands **filter_collection** and **sort_collection** may be called to filter and sort a collection based on some criteria prior to creating a report for the collection elements.

By default, the output file omits the meta data section. If **-metadata** option is present, the output initiates with the meta data section which contains information such as the file generation timestamp, the product name and version and column data types. Some consumers of csv/tsv files benefit from the extra information in the meta data while other consumers do not handle it well.

EXAMPLES

The following example from IC Compiler II writes the full_name and area values of cells in a collection to a comma separated values (CSV) file named "cell_area.db".

```
prompt> set cells [get_cells U*]
prompt> write_collection $cells -columns {full_name area} \
-file cell_area_pins.db
```

The next example outputs the same values to a tab separated values (TSV) file.

```
prompt> write_collection $cells -columns {full_name area} \
-format "tsv" -file cell_area_pins.db
```

The next examples limits the output to a CSV file to the first 10 objects in the collection.

```
prompt> write_collection $cells -columns {full_name area} \
-file cell_area_pins.db -max_rows 10
```

The next example first sorts the original collection of cells by the area attribute value in descending order, limiting the resulting collection to the top 10 area values. The example then outputs the full_name and area values to a CSV file named top_10_areas.db.

```
icc2_shell set sorted_cells [sort_collection -dictionary \
$cells {area- full_name} -limit 10]
prompt> write_collection $cells -columns {full_name area} \
-file top_10_areas.db
```

The next example writes a sorted collection as in the above example but further limits the output to the first 10 objects in the

collection. Note that this command may produce an output that is different from the above example. Limiting the sort to the first 10 values may produce a collection with more than 10 objects since multiple objects may share the same area value.

```
icc2_shell set sorted_cells [sort_collection -dictionary \  
    $cells {area- full_name} -limit 10]  
prompt> write_collection $sorted_cells -columns {full_name area} \  
    -file first_10_objects_by_area.db -max_rows 10
```

The next example outputs to a CSV file which will include the meta data section.

```
prompt> write_collection $cells -columns {full_name area} \  
    -format "csv" -file cell_area_pins.db -metadata
```

SEE ALSO

- collections(2)
- list_attributes(2)
- filter_collection(2)
- sort_collection(2)
- report_collection(2)

write_compile_script

Writes a compile script for the specified design.

SYNTAX

```
int write_compile_script
  [-absolute_paths]
  [-hierarchy]
  [-no_reports]
  [-refining]
  -destination pass
  [-update design_list]
  [design]
```

Data Types

```
pass      string
design_list string
```

ARGUMENTS

-absolute_paths

Uses absolute path names (starting with the ACS working directory, as specified by the **acs_work_dir** variable) when specifying files within the compile script. By default, this command uses relative path names.

-hierarchy

Writes a compile script for each subdesign in the hierarchy that does not have a **dont_touch** attribute. By default, this command writes a compile script only for the specified design.

-no_reports

Specifies not to generate reports. By default, the compile script generates the following reports:

- timing (report_timing)
- area (report_area)
- constraints (report_constraints)
- qor (report_qor)

-refining

Specifies that the compile script is for an incremental compile, rather than a full compile. By default, the compile script is written

for a full compile.

-destination *pass*

Specifies which ACS pass you are in. The command generates the subdirectory name using this value.

-update *design_list*

Specifies the designs to be updated in the already mapped design.

design

Specifies the design to generate a compile script for. You can specify only one design. An error occurs if the design does not exist in Design Compiler memory. If you do not specify a design, ACS uses the current design.

DESCRIPTION

Writes the following files to the *passn/scripts* directory under the ACS working directory (as specified by the *acs_work_dir* variable):

- *env* (This file contains the design environment.)
- *design.suffix* (This file contains the compile script.)

The **acs_compile_script_suffix** variable determines the *suffix* value. By default, the suffix is "autoscr".

If the data directories for the specified pass do not exist, this command creates them.

Automated Chip Synthesis supports the following options for compile script generation. These options are listed below in priority order:

1. Use a user-defined partition compile script. The user-defined partition compile scripts must be located in *scripts/passn/partition_name.scr*.
2. Use the default compile script with a user-defined partition compile. The user-defined partition compile strategies must be located in *scripts/passn/partition_name.compile*.
3. Use the default compile script with a user-defined default compile. The user-defined default compile strategy must be located in *scripts/passn/default.compile*.
4. Use the default compile script with the compile strategy indicated by the ACS compile attributes on the compile partitions. See the **acs_compile_attributes** man page for information about the ACS compile attributes.

The default compile script performs the following tasks:

1. Sets the design environment.
2. Reads the pre-compile .db file for the specified design.
3. For each subdesign,
 1. Reads the post-compile .db file.
 2. Places a *dont_touch* attribute on the subdesign.
4. Sets the current design to the specified design.
5. Links the design.
6. Applies constraints from the file *passn/constraints/design.con*.

7. Compiles the design using the selected compile strategy.
8. Saves all modified designs (top-level and non-dont_touched subdesigns).
9. Generates reports (optional).

The default compile strategy is:

```
# initial compile
set_cost_priority -delay
set_flatten false
compile -map_effort medium

# incremental compile
compile_limit_down_sizing = true
compile -map_effort high -incr
```

SEE ALSO

```
current_design(2)
acs_compile_attributes(3)
acs_compile_script_suffix(3)
acs_constraint_file_suffix(3)
acs_global_user_compile_strategy_script(3)
acs_override_script_suffix(3)
acs_user_compile_strategy_script_suffix(3)
acs_work_dir(3)
```

write_def

Writes the physical data of the specified design to a file in DEF format. To ensure that the names match in the DEF and Verilog files, run the **change_names -rules verilog** command before saving the design.

SYNTAX

```
status write_def
  -output output_file_name
  [-version def_version]
  [-unit conversion_factor]
  [-compressed]
  [-rows_tracks_gcells]
  [-vias]
  [-all_vias]
  [-nondefault_rule]
  [-lef lef_file_name]
  [-regions_groups]
  [-components]
  [-macro]
  [-fixed]
  [-placed]
  [-pins]
  [-blockages]
  [-specialnets]
  [-notch_gap]
  [-pg_metal_fill]
  [-nets]
  [-routed_nets]
  [-diode_pins]
  [-floating_metal_fill]
  [-scanchain]
  [-no_legalize]
  [-verbose]
```

Data Types

```
output_file_name  string
def_version       real
conversion_factor integer
lef_file_name     string
```

ARGUMENTS

-output *output_file_name*

Specifies the name of the DEF file written by the **write_def** command. This is a required argument.

-version *def_version*

Specifies the version of DEF to be written. Valid *def_version* values are 5.3, 5.4, 5.5, 5.6, 5.7 and 5.8.

Note: The Design Compiler tool writes out DEF files in versions 5.7 and 5.8 only, irrespective of this option setting other than 5.7 and 5.8. The default is 5.8.

-unit *conversion_factor*

Specifies the value used in the DEF UNITS DISTANCE MICRONS statement of the DEF file. Valid values for conversion factor are 100, 200, 400, 800, 1000, 2000, 4000, 8000, 10000 and 20000. The values 10000 and 20000 are supported in DEF 5.6. The values 400, 800, 4000 8000 are only supported in DEF 5.8.

If this option is not used, the **write_def** command first checks if the .db representation stores a unit value from a previously read DEF file. If not, the default is 1000.

-compressed

Writes a compressed DEF file in gzip format.

-rows_tracks_gcells

Includes the ROW, TRACK and GCELLGRID sections in the output DEF file. This option can be combined with options for writing out other DEF sections.

-vias

Includes the VIAS section in the output DEF file. This option can be combined with options for writing out other DEF sections.

-all_vias

Includes all vias in the technology file, as well as those present in the design, in the output DEF file. By default, only vias present in the design are written out.

-nondefault_rule

Includes the NONDEFAULTRULES section in the output DEF file, version 5.6 and later. (The NONDEFAULTRULES syntax is not supported in DEF version 5.5 and earlier.)

-lef *lef_file_name*

Writes the rotated vias and design-specified nondefault rule to the specified LEF file.

-regions_groups

Includes the REGIONS and GROUPS sections in the output DEF file. This option can be combined with options for writing out other DEF sections.

-components

Includes the COMPONENTS section in the output DEF file. This option can be combined with options for writing out other DEF sections.

-macro

Includes only macro cells in COMPONENTS section; excludes standard cells. By default, both macro cells and standard cells are included in the COMPONENTS section.

-fixed

Includes only fixed cells in the COMPONENTS section. By default, all fixed, cover, placed, and unplaced cells are included in the

COMPONENTS section.

-placed

Includes only fixed and placed cells in the COMPONENTS section. By default, all fixed, cover, placed, and unplaced cells are included in the COMPONENTS section.

-pins

Includes the PINS section in the output DEF file. This option can be combined with options for writing out other DEF sections.

-blockages

Includes the BLOCKAGES section in the output DEF file. This option can be combined with options for writing out other DEF sections.

-specialnets

Includes the SPECIALNETS section in the output DEF file. This option can be combined with options for writing out other DEF sections. To include notch/gap and PG metal fill as well, use the "-notch_gap" and "-pg_metal_fill" options.

-notch_gap

Includes notch/gap in the SPECIALNETS section.

-pg_metal_fill

Includes PG metal fill in SPECIALNETS section.

-nets

Includes the NETS section in the output DEF file.

-routed_nets

Includes only routed nets in the SPECIALNETS and NETS sections. By default, all nets are included in the SPECIALNETS and NETS sections.

-diode_pins

Includes the diode (extra) pins in the NETS section.

-floating_metal_fill

Includes floating metal fill in the FILLS section.

-scanchain

Includes the SCANCHAINS section in the output DEF file. This option can be combined with options for writing out other DEF sections.

-no_legalize

Prevents name legalization from being performed. Name legalization escapes DEF special characters that do not actually have special meaning. Escaping the special character causes its special meaning to be ignored by the DEF reader. For example, if a component name contains bus bit characters, the bus bit characters are escaped because it is not really a bus. By default, legalization is performed.

-verbose

Reports warning and information messages, in addition to error messages. By default, only error messages are reported.

DESCRIPTION

The **write_def** command writes the physical design data for the specified design in memory to a file in DEF format.

The **write_def** command uses the current design from the open Milkyway design library. You need to open the Milkyway design before you use the **write_def** command.

By default, the command writes out all sections of the DEF file. However, if you use any one or more of the following options, the command writes out only the specified sections: **-rows_tracks_gcells**, **-vias**, **-regions_groups**, **-components**, **-macro**, **-fixed**, **-placed**, **-pins**, **-blockages**, **-specialnets**, **-nondefault_rule**, **-nets**, **-routed_nets**, **-diode_pins**, **-scanchain**, **-notch_gap**, **-floating_metal_fill**, and **-pg_metal_fill**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example writes out the file "out.def" from the currently open Milkyway design.

```
prompt> write_def -lib my_lib -topdesign my_top_design -output out.def
```

SEE ALSO

[open_mw_lib\(2\)](#)

write_design_lib_paths

Writes into a file the paths to which design libraries are mapped.

SYNTAX

```
status write_design_lib_paths  
  [-filename file_name]  
  [-dc_setup]
```

Data Types

file_name string

ARGUMENTS

-filename *file_name*

Specifies the file to which design library information is to be written. If **-dc_setup** is specified, the default file to which the data is written is **.synopsys_dc.setup** in the current working directory. If **-dc_setup** is not specified, the default file to which the data is written is determined by setting the **design_library_file** variable.

-dc_setup

Indicates that the *file_name* is to be written in **.synopsys_dc.setup** format. By default, the file is written in **.synopsys_vss.setup** format (the format used by the Synopsys simulator). If **-dc_setup** is specified, the file written contains **define_design_lib** commands. The **-dc_setup** option changes the default *file_name* value to **.synopsys_dc.setup** in the current working directory.

DESCRIPTION

This command writes into a file the paths to which design libraries are mapped. The format is either **.synopsys_vss.setup** or **.synopsys_dc.setup**. The **write_design_lib_paths** command does not modify an existing file. If the file to which paths are written already exists, **write_design_lib_paths** fails.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, **write_design_lib_paths** writes the current design library paths:

```
prompt> write_design_lib_paths -file ~/.synopsys_dc.setup -dc_setup
```

SEE ALSO

- analyze(2)
- define_design_lib(2)
- elaborate(2)
- get_design_lib_path(2)
- read_file(2)
- report_design_lib(2)

write_environment

Writes the variable settings and constraints for the specified cells or designs.

SYNTAX

```
return_val write_environment
  [-cells cell_list | -designs design_list]
  [-format dcsh | dctl]
  [-output file_name]
  [-suffix suffix]
  [-environment_only]
  [-constraints_only]
  [-no_lib_info]
  [-compress]
  [-consistency]
  [-script]
```

Data Types

```
cell_list    string
design_list   string
file_name    string
suffix       string
```

ARGUMENTS

-cells *cell_list*

The list of instances for which the environment is to be created. This option and the **-designs** option are mutually exclusive. You can specify one or the other or neither. If you specify neither, the command writes the environment for the current design.

-designs *design_list*

The list of designs for which the environment is to be created. This option and the **-cells** option are mutually exclusive. You can specify one or the other or neither. If you specify neither, the command writes the environment for the current design.

-format dcsh | dctl

Specifies the format for the constraints file.

If you do not specify a format, the command uses Tcl format.

-output *file_name*

Specifies the output file name. If you specify the file name, this command generates a single file that contains the requested information. If you do not specify an output file name, this command creates an output file for each partition named

partition_name.suffix, where the suffix value is determined by the **-suffix** option. When the *file_name* ends with the *.gz* extension, the output file will be written out in gzip format.

-suffix *suffix*

Specifies the output file suffix used by this command when you do not specify the file name. The default suffix is "env."

-environment_only

Writes out only the variable settings. By default, both the variable settings and the constraints are written.

-constraints_only

Writes out only the constraints. By default, both the variable settings and constraints are written.

-no_lib_info

Specifies not to include attributes that come from library cells. By default, instances inherit attributes from their reference library cells.

-compress

Writes a compressed output file in gzip format.

-consistency

Writes out variables, attributes, constraints, and so on, for consistency checking. When this option is specified, the **-output** option needs to be specified also.

-script

Writes out the IC Compiler environment in a setup script in Design Compiler in topographical mode. The script includes variables settings, attributes, constraints, and so on. The script ensures that the most updated physical environment is used for additional Design Compiler Graphical runs and helps ensure good correlation of Design Compiler Graphical to IC Compiler post-placement results.

When the **-script** option is specified, the **-output** option must be specified also.

The **-consistency** and **-script** options are mutually exclusive; you can use only one.

DESCRIPTION

For each specified design, this command writes out the following information:

- Settings for all variables whose settings have changed since the session began
- Design constraints
- Attributes inherited from the reference library cells

You can exclude information using the **-constraints_only**, **-environment_only**, and **-no_lib_info** options.

By default, an output file is generated for each design. You can generate a single output file by using the **-output** option.

Multicorner-Multimode Support

This command uses information from the current scenario only.

SEE ALSO

`derive_constraints(2)`

write_file

Writes a design netlist or schematic from memory to a file.

SYNTAX

```
int write_file  
  [-format output_format]  
  [-hierarchy]  
  [-no_implicit]  
  [-output output_file_name]  
  [-scenarios scenario_list]  
  [-library library_name]  
  [-include_anchor_cells]  
  [-exclude_references reference_names]  
  [-pg]  
  [design_list]
```

Data Types

<i>output_format</i>	string
<i>output_file_name</i>	string
<i>scenario_list</i>	list
<i>library_name</i>	string
<i>reference_names</i>	list
<i>design_list</i>	list

ARGUMENTS

-format *output_format*

Specifies the output format of the design. Supported output formats and their descriptions are as follows:

- ddc** - Synopsys internal database format (the default format)
- verilog** - IEEE Standard Verilog
- svsim** - SystemVerilog netlist wrapper
- vhdl** - IEEE Standard VHDL

Specifying **-format svsim** causes the tool to write out only the netlist wrapper, not the gate-level design under test (DUT). To write out the gate-level DUT use **-format verilog**. For further information see the *SystemVerilog User Guide*.

-hierarchy

Writes out all the designs in the hierarchy, starting from the designs specified in *design_list*. If *design_list* is not specified, the current design's hierarchy is written. If a design hierarchy is not completely linked, the unresolved design references are not written out.

If the output format is ddc, block abstractions are not written out unless *design_list* consists of a single design that is the top of the block abstraction's hierarchy. Alternatively, *design_list* can be left empty if the current design is the top of the block abstraction's hierarchy. The **-hierarchy** and **-output** options are required when writing block abstractions to a .ddc file.

-no_implicit

Specifies not to write (save) the synthetic design hierarchy that is implicitly created during the optimization of the design. By default, the command writes designs created in the hierarchy through the use of synthetic libraries even if you do not use the **-hierarchy** option.

-output *output_file_name*

Specifies a single file into which designs are to be written. By default, the command writes each design into a separate file named *design.suffix*, where *design* is the name of each design (with a full UNIX path) and *suffix* is the default suffix for the specified format. The default format suffixes and their descriptions are as follows:

```
.ddc - ddc
.v - verilog
.sv - svsim
.vhd - vhdl
```

The **-output** option is required if the output format is .ddc and a block abstraction hierarchy is being written.

-scenarios *scenario_list*

Lists scenarios whose constraints should be included in the output file. This option applies to ddc format only. By default, the command includes all scenarios.

-library *library_name*

Writes the Synopsys database format object to the specified library.

-include_anchor_cells

Includes anchor cells created during linking (if any) when generating Verilog output. These are, by default, filtered out in Verilog output. This option is supported only by DC Explorer.

-exclude_references *reference_names*

Specifies the reference cell names, separated by space characters, for which all cell instances must not be written. The **-exclude_references** option can only be used with the **-format verilog** option.

-pg

Includes PG nets and ports in Verilog output. If the RTL was read with the **dc_allow_rtl_pg** variable set to **true**, the data obtained from the original RTL is used. Otherwise, PG connections will be inferred from the design's UPF specification, if present. For more information, see the "Instantiating RTL PG Pins in Non-UPF Designs" section in the "Analyzing and Resolving Design Problems" chapter in the *Design Compiler User Guide*. The **-pg** option can only be used with the **-format verilog** option.

design_list

Lists the designs or design files to be written. If *design_list* is omitted, the current design is used as the design list.

Block abstractions are only written to .ddc files when exactly one design is specified, or the current design is inferred, and that design is the top design of a block abstraction hierarchy.

For other output formats, such as .v, if both block abstractions and non-block abstractions exist in the design list, only the non-block abstractions are written out. However, if only block abstractions exist in the design list, the block abstractions are written out.

Designs are not removed from memory after they have been written; to remove a design from memory, use the **remove_design** command.

DESCRIPTION

This command saves the designs from memory to disk. If a design is modified in the tool, you must use the **write_file** command to save the design.

Multicorner-Multimode Support

When writing in ddc format, constraint data for all scenarios is saved by default. You can use the **-scenarios** option to specify a list of scenarios that should be saved.

EXAMPLES

The following examples show how to save designs in ddc format.

This example shows how to write out all designs in the current hierarchy. Since no output file is specified, each design is written to a separate file using the design name as the base of the file name:

```
prompt> write_file -hierarchy
Writing ddc file 'top.ddc'
Writing ddc file 'mid.ddc'
Writing ddc file 'bot.ddc'
```

This example shows how to save all designs in the current hierarchy to a single file:

```
prompt> write_file -hierarchy -output ~bill/dc/designs.ddc
Writing ddc file '/home/bill/dc/designs.ddc'
```

This example shows how to specify a list of designs to be saved:

```
prompt> write_file {ADDER MULT16}
Writing ddc file 'ADDER.ddc'
Writing ddc file 'MULT16.ddc'
```

This example shows another use of the **-hierarchy** option. All designs in the hierarchies of designs A and B are written to a single file:

```
prompt> write_file -hierarchy -output ~bill/dc/two_hiers.ddc {A B}
Writing ddc file '/home/bill/dc/two_hiers.ddc'
```

The following example shows how to specify designs by listing the design file name instead of the design names themselves. (Use **list_designs -show_file** to see the design names associated with each file.) It writes all designs that are associated with the top.ddc file.

```
prompt> write_file top.ddc
Writing ddc file 'top.ddc'.
Writing ddc file 'mid.ddc'.
Writing ddc file 'bot.ddc'.
```

The following example shows how to resolve an ambiguous file name. It is possible to have multiple designs in memory with the same name if they are associated with different files. To resolve the ambiguity you must specify the design in the form *file_name:design_name*. Use the absolute path to the file to avoid possible file name conflicts. In the following example, the design name "top" is ambiguous, and the ambiguity is resolved by using the file name:

```
prompt> write_file -output new.ddc top  
Error: 'top' doesn't specify a unique design  
Please use complete specification: full_file_name:design_name (UID-13)  
prompt> write_file -output new.ddc /home/bill/dc/top.ddc:top  
Writing ddc file 'new.ddc'
```

SEE ALSO

```
read_file(2)  
list_designs(2)  
change_names(2)  
define_name_rules(2)  
create_block_abstraction(2)  
view_write_file_suffix(3)  
dc_allow_rtl_pg(3)
```

write_floorplan

Writes a Tcl script that contains detailed floorplanning information for the specified design. This file can be used to re-create elements of the floorplan.

SYNTAX

```
status write_floorplan
  [-all]
  [-create_bound]
  [-no_bound]
  [-no_create_boundary]
  [-no_placement_blockage]
  [-no_route_guide]
  [-no_voltage_area]
  [-placement {placement_info_types} [-create_terminal]]
  [-preroute]
  [-user_shape]
  [-net_shape]
  [-row]
  [-pin_guide]
  [-track]
  [-def_version def_version]
  [-def_units conversion_factor]
  file_name
  [-format format]
```

Data Types

```
placement_info_types  keyword
def_version           real
conversion_factor     integer
file_name             string
```

ARGUMENTS

-all

Writes all floorplan information to the Tcl script. You can combine the **-all** option with the **-no_bound**, **-no_create_boundary**, **-no_placement_blockage**, **-no_route_guide**, and **-no_voltage_area** options to prevent writing certain types of information. The **-create_terminal** option is ignored and always enabled when using the **-all** option. The **-all** option writes user and net shapes on metal layers that are written by the **-user_shape** and **-net_shape** options.

-create_bound

Writes **create_bounds** commands instead of **update_bounds** commands for bound objects. This option has no effect if you specify the **-no_bound** option. By default, the command writes **update_bounds** commands to the Tcl script.

-no_bound

Prevents the writing of bound information to the Tcl script. By default, bound information is written.

-no_create_boundary

Prevents the writing of boundary information to the Tcl script. By default, the command writes boundary information.

-no_placement_blockage

Prevents the writing of placement blockage information to the Tcl script. By default, the command writes placement blockage information.

-no_route_guide

Prevents the writing of route guide information to the Tcl script. By default, the command writes route guide information.

-no_voltage_area

Prevents the writing of voltage area information to the Tcl script. By default, the command writes voltage area information.

-placement {*placement_info_types*}

Writes the specified placement information to the output file. Valid placement values are: **io**, **hard_macro**, **soft_macro**, and **terminal**. You can specify one or more of these values. The **io** keyword causes the command to write I/O pads. The **soft_macro** keyword causes the command to write black box cells because they are considered soft macros. The **hard_macro** keyword causes the command to write hard macro placements.

The **terminal** keyword causes the command to write pins. When you use the **-placement terminal** option, you should also use the **-create_terminal** option.

By default, the command does not write placement information.

-create_terminal

Writes **remove_terminal** and **create_terminal** commands to the Tcl script. The command writes **create_terminal** commands for terminals if there are pins in the design.

To use this option, you must also use the **-placement {terminal}** option.

-preroute

Writes preroute information to the Tcl script. By default, the command does not write preroute information.

-user_shape

Writes the user shapes created on metal layers by the **create_user_shape** command to the Tcl script. By default, the command does not write these user shapes.

-net_shape

Writes the net shapes created on metal layers by the **create_net_shape** command to the Tcl script. By default, the command does not write these net shapes.

-row

Writes row information to the Tcl script. By default, the command does not write row information.

-pin_guide

Writes pin guide information to the Tcl script. By default, the command does not write pin guide information.

-track

Writes track information to the Tcl script. By default, the command does not write track information.

-def_version *def_version*

Specifies the version of DEF to be written. This option is valid only with "-format icc2". Valid *def_version* values are 5.3, 5.4, 5.5, 5.6, 5.7 and 5.8. The default is 5.8.

Note: The Design Compiler tool writes out DEF files in versions 5.7 and 5.8 only, irrespective of this option setting other than 5.7 and 5.8.

-def_units *conversion_factor*

Specifies the value used in the DEF UNITS DISTANCE MICRONS statement of the DEF file. This option is valid only with "-format icc2". Valid values for conversion factor are 100, 200, 400, 800, 1000, 2000, 4000, 8000, 10000 and 20000. The values 400, 800, 4000 and 8000 are only supported in DEF 5.8.

file_name

Specifies the name of the file to which the floorplan information is written.

-format *format*

Enables writing floorplan files in IC Compiler II format.

By default, the **write_floorplan** command writes floorplan files in Design Compiler format.

When you specify the **-format icc2** option, all the other **write_floorplan** command options are ignored.

DESCRIPTION

This command writes a Tcl script file that contains floorplan information for the current or user-specified design. You restore the floorplan information by reading the generated file with the **read_floorplan** command.

The **write_floorplan** command writes commands relative to the top of the design, regardless of the current instance. The generated Tcl script must be read from the top of the design.

By default, the command writes floorplan information only for the top-level design. The information includes the boundary data and information on the following objects: bounds, placement blockages, route guides, and voltage areas.

When you specify the **-format icc2** option, the **write_floorplan** command writes out the floorplan files in Tcl and DEF format into a folder named by the *file_name* argument. The folder contains a main floorplan.tcl script file, which you can use to load the floorplan in IC Compiler II. The script

- Uses the **read_def** command to load the floorplan.def file
- Sources commands to read the generated Tcl format files
- Uses the remaining settings or constraints that are not captured in the DEF or individual generated Tcl files

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following commands validate the name of the current design, write the floorplan to a file named top.fp, and restore the floorplan by reading it.

```
prompt> write_floorplan -placement {io std_cell hard_macro} \  
        -row top.fp  
1  
prompt> read_floorplan top.fp
```

The following example writes out a floorplan in IC Compiler II format into the folder named my_fp.

```
prompt> write_floorplan -format icc2 my_fp  
1
```

This floorplan information is restored in IC Compiler II by sourcing the floorplan.tcl file.

```
prompt> source my_fp/floorplan.tcl
```

SEE ALSO

read_floorplan(2)

write_icc2_files

Writes the files needed to load the design in IC Compiler II.

SYNTAX

```
status write_icc2_files  
-output dir_name  
[-force]  
[-golden_upf upf_files]  
[-pg]  
[-environment_only]  
[-scenarios scenario_list]  
[-golden_floorplan golden_floorplan_file]  
[-def_version def_version]  
[-def_units conversion_factor]
```

Data Types

<i>dir_name</i>	string
<i>upf_files</i>	list
<i>scenario_list</i>	list
<i>golden_floorplan_file</i>	string
<i>def_version</i>	real
<i>conversion_factor</i>	integer

ARGUMENTS

-output *dir_name*

Specifies the name of the directory in which to write the design and script files.

-force

Overwrites the existing directory. By default, the command fails if the directory already exists or it has no permission to create directory.

-golden_upf *upf_files*

Specifies the golden UPF file that is used in IC Compiler II. This is a required option only in the golden UPF flow, and this options is required to not use in the non-golden UPF flow.

-pg

Includes PG nets and ports in the Verilog output.

-environment_only

Only the *design_name*.setting.tcl file is written in the output directory.

-scenarios *scenario_list*

Specifies a list of scenarios to save. By default, the command includes all scenarios.

-golden_floorplan *golden_floorplan_file*

Specifies the name of the golden floorplan file to use.

-def_version *def_version*

Specifies the version of DEF to be written. Valid *def_version* values are 5.3, 5.4, 5.5, 5.6, 5.7 and 5.8. The default is 5.8.

Note: The Design Compiler tool writes out DEF files in versions 5.7 and 5.8 only, irrespective of this option setting other than 5.7 and 5.8.

-def_units *conversion_factor*

Specifies the value used in the DEF UNITS DISTANCE MICRONS statement of the DEF file. Valid values for conversion factor are 100, 200, 400, 800, 1000, 2000, 4000, 8000, 10000 and 20000. The values 400, 800, 4000 and 8000 are only supported in DEF 5.8.

DESCRIPTION

The **write_icc2_files** command writes the design files and Tcl scripts for the current design to load the design data into IC Compiler II. Files are written into the directory specified by the **-output** option. The written files include the following, if the data is present:

- Netlist in Verilog format
- Propagate backward SAIF
- Tcl floorplan and DEF files
- Design's power intent as a UPF command script
- Scan chain information in SCANDEF format
- Constraints in SDC format
- Timing contexts (scenarios)

In addition, the **write_icc2_files** command writes the following:

- The Design Compiler settings in IC Compiler II format
- The top-level script to load all data generated in IC Compiler II

The **write_icc2_files** command saves the design data for the current design, using the design name to name the generated files.

Use the top-level script, *design_name.icc2_script.tcl* to load all the design data. The script loads design data in the correct order.

The **write_icc2_files** command writes a directory, *design_name*.MCM, which contains timing contexts (scenarios) in a format that can be read into IC Compiler II to create and populate the same set of scenarios as specified in Design Compiler.

In the golden UPF mode, the command writes out the supplemental UPF file and the name mapping file. The top-level script includes load_upf commands to load the golden files specified by the **-golden_upf** option. If the **-pg** option is not specified, in the golden UPF mode, the supplemental UPF file generated by **write_icc2_files** will contain exception power connections derived by

the tool.

The scripts generated by the **write_icc2_files** command do not include scripts for library creation in IC Compiler II. You need to create a library that is consistent with the library used in Design Compiler and then source the top-level script. In the same way, constraints set on the library cell are not included. You need to apply them in IC Compiler II.

When used in conjunction with the **set_host_options** command, the **write_icc2_files** command uses up to the user-specified number of CPU cores on the same computer for parallel execution. See the description of the **-max_cores** option in the **set_host_options** man page for more information.

EXAMPLES

The following example writes the design files to be loaded in IC Compiler II in the `icc2_files` directory for the current design:

```
prompt> write_icc2_files -output icc2_files
```

Next, in IC Compiler II, use the `design_name.icc2_script.tcl` file to load all the design files, after creating the library

```
prompt> source icc2_files/my_design.icc2_script.tcl
```

SEE ALSO

- `write_floorplan(2)`
- `write_file(2)`
- `save_upf(2)`
- `write_saif(2)`
- `write_scan_def(2)`
- `enable_golden_upf(3)`

write_lib

Writes a compiled library to disk as a Synopsys .db file.

SYNTAX

```
int write_lib  
    library_name  
    [-format db]  
    [-output file_name]  
    [-names_file file_list]
```

Data Types

```
library_name  string  
file_name     string
```

ARGUMENTS

library_name

In the Synopsys .db file format, the *library_name* argument specifies the name of the technology or symbol library to be written.

-format db

Specifies the external format of the specified library. Only the .db file format is supported.

-output *file_name*

Specifies an output file name or path name for the library.

DESCRIPTION

The **write_lib** command saves a compiled technology or symbol library to the disk.

If you specify the **-output** option, the command writes out the .db file using the specified path and file name. By default, the command writes out the .db file in the current directory as *library_name*.db.

Note: This command overwrites any existing file without issuing a warning.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to read and write a logic library.

```
prompt> read_lib my_lib.lib
prompt> write_lib my_lib
```

The following example shows how to write a logic library to a specified file.

```
prompt> write_lib my_lib -output ~synopsys/test/test.db
```

The following example shows a name-mapping file called lib.names, and its usage.

Design	Type	Old Name	New Name

lib	reference	NR2	nr2
NR2	pin A	a	
NR2	pin B	b	
NR2	pin Z	z	
lib	reference	HO22	aoi2
HO22	pin I1	a	
HO22	pin I2	b	
HO22	pin I3	c	
HO22	pin O1	z	

SEE ALSO

compare_lib(2)
read_lib(2)

write_lib_specification_model

Writes the PG pin library conversion-specification commands in the output file.

SYNTAX

```
status write_lib_specification_model  
      output_file_name
```

ARGUMENTS

output_file_name

Specifies the output file in which all the commands have to be written out.

DESCRIPTION

The **write_lib_specification_model** command writes out the library update commands and specification commands to the output file. The output file is in Tcl format. This command also writes the **update_lib_model** command to the output file. When the FRAM reference library is not available, the output file written by this command can be sourced to generate a PG pin library.

EXAMPLES

The following command writes the library conversion specification commands to a file named library.tcl.

```
prompt> write_lib_specification_model library.tcl
```

SEE ALSO

```
update_lib_voltage_model(2)  
update_lib_pg_pin_model(2)  
update_lib_pin_model(2)  
update_lib_model(2)  
set_voltage_model(2)  
set_pg_pin_model(2)
```



```
set_pin_model(2)
set_always_on_cell(2)
set_retention_cell(2)
set_level_shifter_cell(2)
set_isolation_cell(2)
set_power_switch_cell(2)
```

write_link_library

Writes shell commands to save the current link library settings for design instances.

SYNTAX

```
int write_link_library
    [-full_path_lib_names] [-nosplit]
    [-full_path_lib_names] [-nosplit]
    [-output file_name]
    [-target target]
```

Data Types

```
file_name  string
target     string
```

ARGUMENTS

-full_path_lib_names

Writes library names with full pathnames. The default is to write only the library name without the path.

-nosplit

Indicates that lines are not to be split when column fields overflow. This is most useful for diffing on previous scripts or for post-processing the script.

-output *file_name*

Writes the script to the specified file. By default, the command writes shell commands to standard output.

-target *target*

Specifies the target to write out scripts. The only valid value is **ptsh**.

DESCRIPTION

This command writes a script of shell commands. It is used to recreate link library settings for each instance in the design.

The redirection operator (>) can be used to redirect the output of this command to a disk.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows output redirected to a file named *output.tcl* and uses the full pathname for output for the link libraries.

```
prompt> write_link_library -output output.tcl \  
-full_path_lib_names
```

SEE ALSO

link_library(3)

write_makefile

Writes a makefile that defines the dependencies and commands required to compile the specified design.

SYNTAX

```
int write_makefile  
  [-absolute_paths]  
  [-dependencies depends]  
  -destination pass  
  [-target target_name]  
  [-lsf [-bsubargs bsub_args]]  
  [-update design_list]  
  [design]
```

Data Types

<i>depends</i>	string
<i>pass</i>	string
<i>target_name</i>	string
<i>bsub_args</i>	string
<i>design_list</i>	string
<i>design</i>	string

ARGUMENTS

-absolute_paths

Uses absolute path names (starting with the ACS working directory, as specified by the **acs_work_dir variable**) when specifying files within the makefile. By default, this command uses relative path names.

-dependencies *depends*

Specifies the dependencies for creating a design database. By default, the design database depends on the post-compile design databases for each compile partition in its hierarchy, the design's constraint file, and the design's compile script.

-destination *pass*

Specifies which ACS pass you are in. The command generates the subdirectory name using this value.

-target *target_name*

Specifies the makefile target. By default, the target is "all".

-lsf

Specifies that the makefile will be run using the Load Sharing Facility (LSF) bsub command when running gmake. By default, the

makefile is created to run under the UNIX make or gmake command.

-bsubargs *bsub_args*

Specifies the arguments for the LSF bsub command. These arguments are passed directly to the LSF software. Automated Chip Synthesis does not validate these arguments. This argument is valid only when used with the **-lsf** option.

-update *design_list*

Specifies the designs to be updated in the already mapped design.

design

Specifies the design to generate a makefile for. You can specify only one design. An error occurs if the design does not exist in Design Compiler memory. If you do not specify a design, ACS uses the current design.

DESCRIPTION

Creates a makefile that defines how to compile the specified design. A makefile defines the dependencies and commands required to build a target (in this case, compile a design). The makefile is written to *passn/file_name* under the ACS working directory (as specified by the **acs_work_dir** variable). The **acs_makefile_name** variable determines the *file_name* value. By default, the file name is "Makefile".

The makefile reads the .db files for each design (compile partitions and the top-level design) from the *passn/db/pre_compile* directory and writes the resulting design to the *passn/db/post_compile* directory. The **dont_touch** attribute setting has the following effect:

- If a design has the **dont_touch** attribute set, the makefile copies the design database from the pre-compile directory to the post-compile directory.
- If a design does not have the **dont_touch** attribute set, the makefile compiles the design using its constraints file (*passn/constraints/design.con*) and compile script (*passn/scripts/design.autoscr*).

If the data directories for the specified pass do not exist, this command creates them.

SEE ALSO

current_design(2)
acs_compile_script_suffix(3)
acs_constraint_file_suffix(3)
acs_db_suffix(3)
acs_makefile_name(3)
acs_work_dir(3)

write_milkyway

Writes out the design to Milkyway database format.

SYNTAX

```
status write_milkyway  
-output filename  
[-overwrite]  
[-scenario scenario_list]
```

Data Types

```
filename      string  
scenario_list list
```

ARGUMENTS

-output *filename*

Specifies the design file name to be written. For example, the following design files are under the CEL view in the *design_lib* Milkyway design library:

```
design_lib/CEL/design1_pre_route:1  
design_lib/CEL/design1_pre_route:2  
design_lib/CEL/design1_post_route:1
```

The *design1_pre_route* and *design1_post_route* are the *filename* arguments. The 1 and 2 after the colons (:) are the version numbers of the file names. This is a required argument.

-overwrite

Overwrites the existing version of the design under the CEL view. This option does not increment the version of the existing file name specified in the **-output** option.

-scenario *scenario_list*

Specifies a list of scenarios to save.

DESCRIPTION

The **write_milkyway** command writes the design into the Milkyway database. The command writes the hierarchical netlist and synthesis constraints. If floorplan, placement, and routing data exists, this information is also written into the Milkyway design library.

If the specified Milkyway design library does not exist, the library is created by **write_milkyway**. This command does not write a design in the Milkyway database if the design is not uniquified or not mapped.

The **write_milkyway** command requires that a variable be set to specify where to find the Milkyway design library. The **mw_design_library** variable must be set before using the **write_milkyway** command:

```
prompt> set mw_design_library <string>
```

The directory name can be a relative name or an absolute name. For a single session, only one Milkyway design library can be accessed. You cannot read or write to different Milkyway design libraries.

Internally, this command writes the design to a CEL view in the Milkyway design library. The exception is when the current design is an Interface Logic Model (ILM), in which case this command writes it to the ILM view. The UNIX structure of the sample the Milkyway design library (CEL/ILM view) is as follows:

```
my_design_lib/           (Milkyway design library)
my_design_lib/CEL/       (CEL view)
my_design_lib/CEL/design1_john:1 (John's CEL data for design1)
my_design_lib/CEL/design1_mike:1 (Mike's CEL data for design1)
my_design_lib/ILM/       (ILM view)
my_design_lib/ILM/design1_john:1 (John's ILM data for design1)
```

Assume the **mw_design_library** variable is set to *my_design_lib*. The UNIX file called *my_design_lib/CEL/design1_john:1* is created when you run the following command:

```
prompt> write_milkyway -output design1_john
```

EXAMPLES

In the following example, the contents of the design are written to the *MW_DESIGN_LIB* design library. The saved design file is called *RISC_CORE*.

```
prompt> set mw_design_library MW_DESIGN_LIB
prompt> write_milkyway -output RISC_CORE
```

In the following example, an ILM is created and saved to the ILM view in the *mw_design_library MW_DESIGN_LIB*. The **write_milkyway** command saves to the ILM view because the design in-memory after the **create_ilm** command is an Interface Logic Model.

```
prompt> set mw_design_library MW_DESIGN_LIB
prompt> create_ilm -physical
prompt> write_milkyway -output RISC_CORE
```

SEE ALSO

[mw_design_library\(3\)](#)

write_multibit_components

Writes out a set of **create_multibit** command statements that if executed will allow the user to re-create the current set of multibit components of the design.

SYNTAX

```
status write_multibit_components  
[-output output_file_name]
```

Data Types

output_file_name string

ARGUMENTS

DESCRIPTION

The **write_multibit_components** command allows the user to preserve the multibit components through an ASCII interface when a DDC flow cannot be used and an ASCII flow approach is needed.

The command will output a set of **create_multibit** commands for the user to source them after reading the netlist in ASCII format.

This command is not needed in DDC flow, as the multibit components are preserved in the DDC file.

SEE ALSO

create_multibit(2)
remove_multibit(2)
identify_register_banks(2)

write_multibit_guidance_files

Writes out the mapping guidance files for the input map file and the register group file that can be used for placement-based multibit mapping in Design Compiler Graphical and IC Compiler.

SYNTAX

```
status write_multibit_guidance_files  
[-prefix prefix_of_output_file_names]
```

Data Types

prefix_of_output_file_names string

ARGUMENTS

-prefix *prefix_of_output_file_names*

Specifies the prefix for the mapping guidance file names. The default prefix is multibit_guidance.

DESCRIPTION

The **write_multibit_guidance_files** command identifies the multibit registers that are available in the target and link libraries and the single-bit registers that can be replaced by the multibit registers based on the functional information of the library cells.

The **write_multibit_guidance_files** command uses the same process as the **identify_register_banks** command to identify the multibit and single-bit registers and then generates the following guidance files based on the functional information of the library cells:

- multibit_guidance.input_map (input map file)
 - multibit_guidance.register_group (register group file)

Examine these files to see which multibit cells are available in the library and which single-bit cells can be replaced by multibit cells.

If you do not need to change the way these files define multibit mapping, you do not need to specify the input map file and the register group file when you run the **identify_register_banks** command. In this case, Design Compiler automatically uses the guidance from the files generated by the **write_multibit_guidance_files** command.

To change the way multibit mapping is performed, modify these guidance files and provide additional multibit mapping guidance. You must specify the modified files when you run the **identify_register_banks** command in Design Compiler Graphical.

To use these guidance files in IC Compiler, specify them with the **set_banking_guidance_strategy** command.

See the **identify_register_banks** command man page for the mapping guidance files format.

SEE ALSO

identify_register_banks(2)

write_mw_lib_files

Writes the technology, or plib, or reference control file of the Milkyway library.

SYNTAX

```
status_value write_mw_lib_files
[-technology]
[-reference_control_file]
[-stream_layer_map_file file_format]
-output file_name
[libName]
```

Data Types

<i>file_format</i>	string
<i>file_name</i>	string
<i>libName</i>	string

ARGUMENTS

-technology

Writes the technology information of the specified Milkyway library into a new text file in .tf format. To write the technology information from a Milkyway library other than the one currently open, you must first close the open Milkyway library.

These options are mutually exclusive: **-technology**, **-plib**, **-reference_control_file**, and **-stream_layer_map_file**.

-reference_control_file

Writes the reference control file of the specified Milkyway library into a new text file. The Milkyway reference control file is a list of Milkyway libraries referenced by the design library.

-stream_layer_map_file *file_format*

Writes a layer mapping file of the specified Milkyway library into a new text file. The *file_format* argument must be one of the following:

- * **in** for the stream-in layer mapping file
- * **out** for the stream-out layer mapping file
- * **starrc** for the StarRC layer mapping file
- * **all** for all of the above layer mapping files

-output *file_name*

Specifies the name of the new text file written by the command.

libName

Specifies the name of the Milkyway library from which to obtain the .tf technology information, .plib physical library information, reference control file, or layer mapping information.

DESCRIPTION

Writes a specified type of information obtained from a specified Milkyway library into an ASCII text file.

You must use exactly one of the following options to specify the type of information to write: **-technology**, **-plib**, **-reference_control_file**, or **-stream_layer_map_file**.

The command returns 1 for success or 0 for failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command writes the .tf technology file of the Milkyway library named my_mw_lib. The new text file created is named my_tech.tf.

```
prompt> write_mw_lib_files -technology -output my_tech.tf
prompt> my_mw_lib
Technology file data dumped to my_tech.tf completely.
1
```

The following command writes out the stream-out layer mapping file of the Milkyway library named my_mw_lib. The new text file created is named my_layer_map.out.

```
prompt> write_mw_lib_files -stream_layer_map_file out \
-output my_layer_map.out my_mw_lib
1
```

SEE ALSO

create_mw_lib(2)
set_mw_lib_reference(2)
set_mw_technology_file(2)

write_parasitics

Writes parasitics in SPEF format or as a Tcl script that contains **set_load** and **set_resistance** commands.

SYNTAX

```
status write_parasitics
      [-output file_name]
      [-format reduced]
      [-min]
      [-ratio ratio_number]
      [-script]
```

Data Types

```
file_name    string
ratio_number float
```

ARGUMENTS

-output *file_name*

Specifies the name of the output file to which parasitics for the current design are written.

If you do not specify this option, the parasitics are written to a file named *design_name.format_name*, where *design_name* is the name of the current design.

-format *reduced*

Specifies to write the parasitics in reduced SPEF format.

-min

Writes parasitics for the minimum operating condition. By default, the parasitics are written for the maximum operating conditions.

-ratio *ratio_number*

Specifies the ratio used when writing the pie model description for every driver in the net. The ratio specified by this command must be in the range 0.0 - 1.0. By default, the nets in reduced format are generated with a ratio equal to 0.5. The capacitor closest to the driver is 0.5 (the ratio value) times the total net capacitance in the pie model description.

-script

Writes out **set_load** and **set_resistance** commands instead of an SPEF file.

DESCRIPTION

This command writes parasitics for the current design to a disk file.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following example writes parasitics in reduced SPEF format to a file named top.reduced.

If there are minimum and maximum operating conditions, parasitics for both conditions are written.

```
prompt> write_parasitics -format reduced -output top
```

```
*SPEF 1.0
*DESIGN "filter"
*DATE "Tue May 28 10:01:39 1996"
*VENDOR "SYNOPTICS INC"
*PROGRAM "Synopsys Design Compiler cmos"
*VERSION "3.5a-SI2-SCM"
*DESIGN_FLOW "SYNTHESIS"
*DIVER /
*DELIMITER :
*T_UNIT 1.0 NS
*C_UNIT 1.0 PF
*R_UNIT 1.0 KOHM
*L_UNIT 1.0 HENRY
```

```
*PORTS
```

```
micro_d_1 | *L 0.000 *S 2.980 2.980
micro_d_2 | *L 0.000 *S 2.980 2.980
micro_d_3 | *L 0.000 *S 2.980 2.980
micro_d_4 | *L 0.000 *S 2.980 2.980
micro_d_5 | *L 0.000 *S 2.980 2.980
micro_d_6 | *L 0.000 *S 2.980 2.980
micro_d_7 | *L 0.000 *S 2.980 2.980
micro_d_8 | *L 0.000 *S 2.980 2.980
micro_d_9 | *L 0.000 *S 2.980 2.980
micro_d_10 | *L 0.000 *S 2.980 2.980
micro_d_11 | *L 0.000 *S 2.980 2.980
micro_d_12 | *L 0.000 *S 2.980 2.980
preset[15] | *L 0.000 *S 0.249 0.204 *D IVA
```

```
*D_NET W03[14] 3.350e-02
```

```
*CONN
```

```
*P W03[14] O *L 0.0
```

```
*I U111112:ZN O *L 0.0
```

*CAP

1 W03[14] 1.300e-03

2 U111112:ZN 0.0

3 W03[14]:16 1.300e-03

4 W03[14]:6 1.301e-04

5 W03[14]:14 2.380e-03

6 W03[14]:5 1.301e-04

*RES

1 W03[14] W03[14]:16 1.880e-02

2 U111112:ZN W03[14]:6 1.600e-03

3 W03[14]:16 W03[14]:14 1.600e-03

4 W03[14]:6 W03[14]:5 9.081e-04

5 W03[14]:14 W03[14]:19 2.792e-02

*END

SEE ALSO

`extract_rc(2)`

`read_parasitics(2)`

`set_load(2)`

`set_resistance(2)`

write_partition

Writes the database for a design into the Automated Chip Synthesis data structure.

SYNTAX

```
int write_partition  
  -type pre | post  
  [-destination pass]  
  [-hierarchy]  
  [design]  
  [-format db | ddc]
```

ARGUMENTS

-type pre | post

Specifies which Synopsys database format (.db) file to write. The valid keywords are **pre**, for the precompile .db type, and **post**, for the postcompile .db type. The **pre** and **post** values are mutually exclusive. If you specify the **pre** value, the command writes the .db file in the *pass**n*/db/pre_compile directory. If you specify the **post** value, the command writes the .db file in the *pass**n*/db/post_compile directory.

-destination *pass*

Specifies which Automated Chip Synthesis pass you are in. Substitute the name of the pass you want for *pass*. The command generates the subdirectory name using this value.

-hierarchy

Writes .db files for all compile partitions and master instance reference designs in the hierarchy of the specified design. By default, the command writes .db files for the specified design and for the master instance reference designs.

design

Specifies the design to write. Specify only one design, substituting it for *design_name*. An error occurs if the specified design does not exist in Design Compiler memory. If you do not specify a design, Automated Chip Synthesis uses the current design.

DESCRIPTION

Writes the database for a design into the Automated Chip Synthesis data structure. For use only in dc_shell-t (Tcl mode of dc_shell). The command writes the design database to the file *design.suffix* in the Automated Chip Synthesis directory structure. The **acs_db_suffix** variable determines the *suffix* value. By default, the suffix is *db*. The path for the .db file is determined by the command arguments.

If the data directories for the specified pass do not exist under the Automated Chip Synthesis working directory (as specified by the **acs_work_dir** variable), the command creates them.

SEE ALSO

current_design(2)
read_partition(2)
write(2)
acs_db_suffix(3)
acs_work_dir(3)
MasterInstance(3)

write_partition_constraints

Writes out the timing constraints for a design.

SYNTAX

```
int write_partition_constraints  
  [-hierarchy]  
  -destination pass  
  [-update design_list]  
  [design]
```

ARGUMENTS

-hierarchy

Writes a constraint file for each subdesign in the hierarchy that does not have a dont_touch attribute. By default, this command writes a constraint file only for the specified design.

-destination *pass*

Specifies which Automated Chip Synthesis pass you are in. Substitute the name you want for *pass*. The command generates the subdirectory name using this value.

-update *design_list*

Specifies the designs to be updated in the already-mapped design. Substitute the list you want for *design_list*.

design

Specifies the design to generate constraints for. You can specify only one design, substituting it for *design*. An error occurs if the design does not exist in Design Compiler memory. If you do not specify a design, Automated Chip Synthesis uses the current design.

DESCRIPTION

Writes out the timing constraints for a design. For use only in dc_shell-t (Tcl mode of dc_shell). The command writes the constraints present on the design database to the file `passn/constraints/design.suffix` under the Automated Chip Synthesis working directory (as specified by the **acs_work_dir** variable).

The **acs_constraint_file_suffix** variable determines the suffix value. By default, the value is *con*.

If the data directories for the specified pass do not exist, this command creates them.

SEE ALSO

current_design(2)
write_environment(2)
acs_constraint_file_suffix(3)
acs_work_dir(3)

write_physical_constraints

Writes the script of Tcl commands to export the current physical constraint settings. This command is supported only in topographical mode. Note: The **write_physical_constraints** command is now aliased to the **write_floorplan** command. You should use the **write_floorplan** command instead.

SYNTAX

```
status write_physical_constraints
  -output tcl_file
  [-no_site_row]
  [-pre_route]
```

Data Types

tcl_file string

ARGUMENTS

-output *tcl_file*

Specifies the name of the file to which the physical constraints are written. This argument is required.

-no_site_row

Indicates that site rows commands are not dumped in the output file. By default, site row commands are written out to the output file.

-pre_route

Writes pre-routes to the output file. By default, pre-routes are not written to the output file.

DESCRIPTION

The **write_physical_constraints** command generates a script file containing the Tcl commands used to set the physical constraints that have been applied to the design. The script file is saved in the file specified with the **-output** option.

You can use the generated script file to reapply the physical constraints at a later time by the **read_floorplan** command.

The **write_physical_constraints** command writes commands relative to the top of the design, regardless of the current instance. The output file it writes must be read from the top of the design.

By default, the physical constraints information is written only for the top-level design and includes the boundary information, as well

as the following objects: bounds, placement blockages, route guides, plan groups, and voltage areas.

For the physical constraints of pre-routes, the corresponding command is **create_net_shape**. In order to write out the commands, specify the **-pre_route** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example command writes the physical constraints into the file named script.tcl:

```
prompt> write_physical_constraints -output script.tcl
```

SEE ALSO

extract_physical_constraints(2)
report_physical_constraints(2)

write_qtm_model

Writes the Quick Timing Model (QTM) file.

SYNTAX

```
string write_qtm_model  
  -out_dir output_qtm_directory  
  [-text]
```

Data Types

output_qtm_directory string

ARGUMENTS

-out_dir *output_qtm_directory*

Specifies the name of the output QTM directory

-text

To write out text format file for the QTM model in addition to the .db format file.

DESCRIPTION

This command writes out the QTM model file in Synopsys .db format.

If the *-text* option is provided then the QTM model file is written out in text format also.

This command writes out all the QTM Model(s) in the internal QTM database with different names in one output directory. You must provide the output directory as a command-line option.

The name of the QTM Model is given in **create_qtm_model** command. The name of the output QTM Model file is derived from the QTM Model Name. The output QTM Model file name is *<ModelName>.qtm.db*.

This command writes out the in-memory .db to on-disk .db file. This command runs slower than **save_qtm_model** command and requires disk I/O. Usually, this command is expected to be called once at the end of an ICC session to get a final version of the QTM model.

For a more detailed description about QTM please refer to the *ICC Design Planning User Guide*.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example writes the QTM Model **adder** in internal QTM database in directory **qtm** as **adder.qtm.db**. This will generate a file **./qtm/adder.qtm.db**

```
prompt> write_qtm_model -out_dir qtm
```

SEE ALSO

`create_qtm_model(2)`
`save_qtm_model(2)`

write_rp_groups

Writes out the relative placement constraints for the specified relative placement groups.

SYNTAX

```
collection write_rp_groups  
  rp_groups | -all  
  [-hierarchy]  
  [-quiet]  
  [-nosplit]  
  [-output filename]  
  [-create]  
  [-leaf]  
  [-keepout]  
  [-instance]  
  [-include]
```

Data Types

```
rp_groups    list or collection  
filename    string
```

ARGUMENTS

rp_groups

Specifies the relative placement groups to be written out to the script.

This option is mutually exclusive with the **-all** option.

-all

Specifies that all relative placement groups are to be written out to the generated script.

This option cannot be used with either the *rp_groups* argument or the **-hierarchy** option.

-hierarchy

Specifies that all relative placement groups within the hierarchy of the groups in *rp_groups* are to be written out. By default, subgroups are not written out.

This option is mutually exclusive with the **-all** option.

-quiet

Turns off informational messages that would otherwise be issued if no groups are written.

-nosplit

Specifies not to split long commands into multiple lines. The default is to split long commands into multiple lines.

-output *filename*

Specifies that the script is to be written to *filename*. The default is to write to standard output.

-create

Specifies that the **create_rp_group** commands are included in the generated script.

-leaf

Specifies that the **add_to_rp_group -leaf** commands are included in the generated script.

-keepout

Specifies that the **add_to_rp_group -keepout** commands are included in the generated script.

-instance

Specifies that the **add_to_rp_group -hierarchy -instance** commands are included in the generated script.

-include

Specifies that the **add_to_rp_group -hierarchy** commands (without **-instance**) are included in the generated script.

DESCRIPTION

The **write_rp_groups** command writes the relative placement constraints for the specified relative placement groups. You can use the generated commands to re-create the relative placement groups and their items on the same designs.

When you specify any of the **-create**, **-leaf**, **-keepout**, **-instance**, and **-include** options, the generated script generated contains only the commands related to the specified options. If you do not specify any of these options, all commands are output to the generated script.

The command returns a collection that contains the relative placement groups that are written out. If no objects are written out, the empty string is returned.

Relative placement usage is available with Design Compiler Ultra.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses **write_rp_groups** to re-create relative placement groups after their removal:

```
prompt> get_rp_groups  
{mul::grp_mul ripple::grp_ripple example3::top_group}
```

```
prompt> write_rp_groups -all -output my_groups.tcl  
{mul::grp_mul ripple::grp_ripple example3::top_group}
```

```
prompt> remove_rp_groups -all -quiet  
1
```

```
prompt> get_rp_groups  
Error: Can't find objects matching '*'. (UID-109)
```

```
prompt> source my_groups.tcl  
{example3::top_group}
```

```
prompt> get_rp_groups  
{example3::top_group ripple::grp_ripple mul::grp_mul}
```

SEE ALSO

add_to_rp_group(2)
create_rp_group(2)
remove_rp_groups(2)

write_rtl_load

Writes a script of RTL load commands for the current design.

SYNTAX

```
int write_rtl_load  
  [-format dctl | dcsh]  
  [-output file_name]
```

Data Types

file_name string

ARGUMENTS

-format dctl | dcsh

Specifies that the script is to be written in dctl or dcsh mode. By default, the script is written in the format of the mode from which this command is executed.

-output *file_name*

Specifies that the script is to be written to the specified file. If **-output** is not specified, the script is written to standard output.

DESCRIPTION

The **write_rtl_load** command writes a script of **set_rtl_load commands**. You can use this command to display the RTL loads on the current design, or to write a **set_rtl_load** script file to apply to another design.

By default, the **write_rtl_load** command writes commands to standard output. The **-output** option redirects the output to the specified disk file.

EXAMPLES

In the following example, the RTL loads of the current design are written to the file test.scr.

```
prompt> write_rtl_load -output test.scr
```

In the following example, the RTL loads of the current design are written in dcsb format to standard output.

```
prompt> write_rtl_load
```

SEE ALSO

`set_rtl_load(2)`
`write_script(2)`

write_saif

Writes a backward Switching Activity Interchange Format (SAIF) file.

SYNTAX

```
status write_saif  
-output file_name  
[-instances instances]  
[-no_hierarchy]  
[-rtl]  
[-propagated]  
[-exclude_sdpd]
```

Data Types

```
file_name    string  
instances    list
```

ARGUMENTS

-output *file_name*

Specifies the name of the output SAIF file, if the *file_name* ends with the .gz extension, the output file is written in gzip format.

-instances *instances*

Specifies an optional list of instances for which the SAIF file is generated. If this argument is not specified, then the SAIF file is generated for the current instance.

-no_hierarchy

This flag specifies that the SAIF file contains switching activity information for only the top-level design objects. When this flag is not specified, **write_saif** generates a SAIF file containing switching activity information for all the objects in the hierarchy.

-rtl

Specifies that the generated SAIF file contains the switching activity for the synthesis invariant objects. These are the design ports, hierarchical cell pins, and outputs of sequential and tri-state cells. When this option is used on an unmapped design, the generated SAIF file is similar to one generated by RTL simulation flow using an RTL SAIF file. The generated SAIF file can then be read correctly into the synthesized design. If this option is not used when writing a SAIF file for an unmapped design, then the generated SAIF file might not be read correctly after the design has been synthesized, although it can be read correctly on the unmapped design. Note that the *-rtl_direct* of the **read_saif** command should not be used when reading SAIF files generated by **write_saif**.

-propagated

Propagates the user-annotated switching activity to estimate the switching activity of unannotated objects, and generates a SAIF file containing both the annotated and estimated switching activity. When this option is not specified, the **write_saif** command generates a SAIF file containing the user-annotated switching activity only.

-exclude_sdpg

Excludes state dependent static probabilities and state dependent or path dependent toggle rates from the SAIF file.

DESCRIPTION

The **write_saif** command generates a backward SAIF file containing the design switching activity information.

The **write_saif** command can generate both user-annotated and propagated switching activity information. If you specify the **-propagated** option, the command propagates the switching activity information and generates a SAIF file with both user-annotated and propagated switching activity. If you do not specify the **-propagated** option, the command generates only the user-annotated switching activity.

The top-level instance name in the generated SAIF file is the current design name. This name needs to be specified as the instance name when reading the SAIF file with the **read_saif** command.

When you specify the **-exclude_sdpg** option, the SAIF file contains only the following simple switching activity information: the toggle rate and static probability of the design nets, ports, and cell pins that are not connected to nets.

Note, the related clock information cannot be included in the SAIF file. The toggle rates in the SAIF file are converted to be relative to the synthesis time unit automatically.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example creates a SAIF file containing user-annotated switching activity information on all design objects.

```
prompt> write_saif -output design.saif
```

The following example creates a compressed SAIF file in gzip format containing user-annotated switching activity information on all design objects.

```
prompt> write_saif -output design.saif.gz
```

The following example creates a SAIF file containing both user-annotated and propagated switching activity information.

```
prompt> write_saif -output design.saif -propagated
```

The following example creates a SAIF file on a synthesized design object, and reads in the same file.

```
prompt> write_saif -output design.saif
prompt> reset_switching_activity
prompt> read_saif -input design.saif -instance top
```

The following example generates a SAIF file with user-annotated and propagated switching activity, but does not contain state-dependent and path-dependent information.

```
prompt> write_saif -output design.saif -propagated -exclude_sdpd
```

The following example generates an RTL SAIF file from an unmapped design and read in the file.

```
prompt> write_saif -output rtl.saif -rtl
```

```
prompt> reset_switching_activity
```

```
prompt> compile
```

```
prompt> read_saif -input rtl.saif -instance top
```

SEE ALSO

`set_switching_activity(2)`

`reset_switching_activity(2)`

`read_saif(2)`

`merge_saif(2)`

`report_saif(2)`

write_scan_def

Generates SCANDEF scan chain information for performing scan chain reordering in the physical implementation flow.

SYNTAX

```
int write_scan_def  
  [-output def_file]  
  [-expand_elements list_of_cells]
```

ARGUMENTS

-output *def_file*

Specifies the name of the SCANDEF output file to create for the physical implementation tool. The default name is *current_design.def*, in the current directory.

-expand_elements *list_of_cells*

Specifies a list of CTL-modeled cell instances whose scan segments must be treated as 'flat' fully visible segments when generating the SCANDEF file.

The default behavior of SCANDEF generation is to treat all cell instances with CTL test model information as black boxes, which are represented in the SCANDEF by the "BITS" construct. When a CTL-modeled cell is specified with this option, the scan chain contents of the cell are included in the SCANDEF file.

DESCRIPTION

The **write_scan_def** command generates scan chain information in SCANDEF format. It is a post-DFT command, intended for use after scan chains are inserted into the design.

Reordering of scan chains in physical implementation tools, such as IC Compiler, requires information about the scan chains that exist in the design. This is because there can be aspects of the scan chains, such as lock-up latches, clock-mixing, or user-defined scan paths or scan segments, that must be considered during reordering. The SCANDEF information describes these scan chain ordering requirements.

The **write_scan_def** command writes the SCANDEF information to the file specified by the **-output** option, or to the default file name if the option is not specified. The command also annotates the SCANDEF information to the current design in memory. If you write the design out in .ddc format, the SCANDEF information is included in the .ddc file. In this case, IC Compiler does not need the SCANDEF file written to disk, but you can use the file for reference.

Generating SCANDEF in a Hierarchical DFT Flow

For any DFT-inserted cores in the design that contain CTL test model information, the **write_scan_def** command abstracts the contents of the core cell with the "BITS" construct. This prevents reordering inside the cell, which is desired behavior for hard and soft macros that are frozen during top-level optimization.

To include ("expand") the contents of DFT-inserted cores, specify the list of core instances with the **-expand_elements** option. These cores must themselves contain SCANDEF information, which is accomplished in the core-level run by using the **write_scan_def** command, then writing out the core design in .ddc or .ctlddc format. If a core does not contain SCANDEF information, DFT Compiler issues a warning:

Warning: SCANDEF information for design instance %s is not available. NETLIST information is available. SCANDEF for design instance %s will be expanded using netlist information.

In this case, DFT Compiler expands the core scan chains by exploring the core netlist structure. Basic reordering requirements such as clock mixing are inferred, but any user-applied core-level scan constraints (such as scan group or scan path definitions) are lost.

EXAMPLES

The following example writes out a SCANDEF file named HUNSLEY.def for the current design HUNSLEY, using the default SCANDEF generation options.

```
prompt> current_design HUNSLEY
prompt> create_test_protocol
prompt> dft_drc
prompt> insert_dft
prompt> dft_drc
prompt> write_scan_def
prompt> check_scan_def
```

The following example writes out a SCANDEF file named HUNSLEY.def that includes the scan elements inside cores CORE1 and CORE2.

```
prompt> current_design HUNSLEY
prompt> create_test_protocol
prompt> dft_drc
prompt> insert_dft
prompt> dft_drc
prompt> write_scan_def -output scan.def \
    -expand_elements [list CORE1 CORE2]
prompt> check_scan_def
```

SEE ALSO

`create_test_protocol(2)`
`dft_drc(2)`
`preview_dft(2)`
`insert_dft(2)`
`set_scan_configuration(2)`
`check_scan_def(2)`

write_script

Writes shell commands to save the current settings.

SYNTAX

```
int write_script  
  [-hierarchy]  
  [-no_annotated_check] [-no_annotated_delay] [-no_cg]  
  [-full_path_lib_names] [-nosplit]  
  [-format dctl | dcsh]  
  [-include loop_breaking]  
  [-output file_name]
```

ARGUMENTS

-hierarchy

Writes commands for all designs in the hierarchy.

-no_annotated_check

Indicates that **set_annotated_check** commands are not to be written. By default, annotation commands are written to standard output. Use this option to avoid creating a very large script for designs that contain a large amount of annotated information. Annotated timing checks can be written to a file using the **write_timing** command.

-no_annotated_delay

Indicates that **set_annotated_delay** commands are not to be written. By default, annotation commands are written to standard output. Use this option to avoid creating a very large script for designs that contain a large amount of annotated information. Annotated delays can be written to a file using the **write_timing** command.

-no_cg

Indicates that Power Compiler clock gating attributes are not to be written. By default, **set_attribute** commands are written to standard output for all relevant clock gating attributes. Use this option if clock gate information is not needed for later flow.

-full_path_lib_names

Indicates that library names are to be written with full pathnames. The default is to write only the library name, without the path.

-nosplit

Indicates that lines are not to be split when column fields overflow. This is most useful for doing diffs on previous scripts, or for post-processing the script.

-format dctl | dcsh

Indicates that the script is to be written in Tcl mode or dcsh mode.

-include loop_breaking

Writes the **set_disable_timing** commands for internally disabled timing arcs.

-output file_name

Indicates that the script is to be written to the specified file.

DESCRIPTION

The **write_script** command writes a script of dc_shell commands. This command is used to recreate the attributes on the current design.

By default, the **write_script** command writes dc_shell commands to standard output. User-defined attributes are not supported. Some of the commands whose constraints are written out by **write_script** are listed below. (This is not a complete list.)

```
create_clock
group_path
set_annotated_check
set_annotated_delay
set_boundary_optimization
set_connection_class
set_disable_timing
set_dont_retime
set_dont_touch
set_dont_touch_network
set_drive
set_driving_cell
set_equal
set_false_path
set_fanout_load
set_fix_hold
set_fix_multiple_port_nets
set_flatten
set_implementation
set_input_delay
set_ideal_latency
set_ideal_net
set_ideal_network
set_ideal_transition
set_load
set_local_link_library
set_logic_one
set_logic_zero
set_logic_dc
set_map_only
set_max_area
set_max_delay
set_max_fanout
set_max_power
set_max_time_borrow
set_max_transition
```

```

set_min_delay
set_multicycle_path
set_operating_conditions
set_opposite
set_optimize_registers
set_output_delay
set_port_fanout_number
set_register_type
set_resistance
set_rtl_load
set_signal_type
set_size_only
set_structure
set_switching_activity
set_test_assume
set_test_hold
set_timing_ranges
set_unconnected
set_ungroup
set_wire_load_model
set_wire_load_mode
set_wire_load_selection_group
set_wire_load_min_block_size
set_wired_logic_disable

```

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

In the following example, the settings on the current design are written to the file test.scr.

```
prompt> write_script -output test.scr
```

In the following example, the settings on the current design are written to standard output. An example of the output file is also provided.

```

prompt> write_script
/******

Created by write_script() on Fri Jun 21 17:57:25 1991

*****/

/* Set the current_design */
current_design TOP

set_operating_conditions WCCOM
set_wire_load_model WL_SMALL_BLOCK
create_clock -period 100 -waveform {0 5} [get_ports c1]
set_input_delay 1.5 -clock c1 [get_ports B]
set_input_delay 2.8 -clock c1 [get_ports A]

```

In the following example settings on all the designs in the hierarchy are written to standard output. An example of the output file is also provided.

```
prompt> write_script -hierarchy
```

```
#####
```

```
# Created by write_script -format dctl on Mon Mar 20 07:21:11 2006
```

```
#####
```

```
# Set the current_design #  
current_design top
```

```
remove_wire_load_model  
set_dont_touch [get_cells mid1]
```

```
# Set the current_design #  
current_design mid
```

```
remove_wire_load_model  
set_dont_touch [get_cells low1]
```

```
# Set the current_design #  
current_design low
```

```
remove_wire_load_model  
1
```

SEE ALSO

```
current_design(2)  
reset_design(2)  
write(2)
```

write_sdc

Writes out a script in Synopsys Design Constraints (SDC) format.

SYNTAX

```
int write_sdc  
    file_name  
    [-nosplit]  
    [-version sdc_version]
```

Data Types

file_name string

ARGUMENTS

file_name

Specifies the name of the file to which the SDC script is to be written.

-nosplit

Indicates that lines are not to be split when column fields overflow. This is most useful when comparing previous script files with the UNIX diff command, or for post-processing the script.

-version *sdc_version*

Specifies the version of SDC to write. Allowed values are **1.2**, **1.3**, **1.4**, **1.5**, **1.6**, **1.7**, **1.8**, **1.9**, **2.0**, **2.1** and **latest** (the default).

DESCRIPTION

The **write_sdc** command writes out a script file in the latest Synopsys Design Constraints (SDC) format. This script contains commands that can be used with PrimeTime or with Design Compiler. SDC is also licensed by external vendors through the Tap-in program. SDC-formatted script files are read into PrimeTime or Design Compiler using the **read_sdc** command.

By default, the script is written as simple text. Also, by default the latest version of SDC is written. Some earlier versions of SDC can be written using the **-version** option.

SDC is a subset of the commands already supported by Design Compiler, Physical Compiler, IC Compiler, Astro, Jupiter, and PrimeTime. Of the commands supported in the latest SDC version, the following may be written by **write_sdc**. Those added for versions 1.3 and later are noted.

General Purpose Commands:

- expr
- list
- set

Object Access Functions:

- all_clocks
- all_inputs
- all_outputs
- all_registers (1.7)
- current_design
- current_instance
- get_cells
- get_clocks
- get_libs
- get_lib_cells
- get_lib_pins
- get_nets
- get_pins
- get_ports
- set_hierarchy_separator

Basic Timing Assertions:

- create_clock
- create_generated_clock (1.3)
- group_path (1.7)
- set_clock_gating_check
- set_clock_groups (1.7)
- set_clock_latency
- set_clock_sense (1.7)
- set_clock_transition
- set_clock_uncertainty
- set_false_path
- set_ideal_latency (1.7)
- set_ideal_transition (1.7)
- set_input_delay
- set_max_delay
- set_min_delay
- set_multicycle_path
- set_output_delay
- set_propagated_clock
- set_sense (2.1)

New options to existing supported commands:

- create_clock -add (1.4)
- create_generated_clock -add (1.4)
- create_generated_clock -master_clock (1.4)
- create_generated_clock -combinational (1.7)
- set_clock_latency -dynamic (2.1)
- set_max_delay -ignore_clock_latency (2.1)
- set_min_delay -ignore_clock_latency (2.1)

Secondary Assertions:

```
set_disable_timing
set_max_time_borrow
```

Environment Assertions:

```
create_voltage_area (1.6)
set_case_analysis
set_drive
set_driving_cell
set_fanout_load
set_input_transition
set_ideal_network (1.7)
set_level_shifter_strategy (1.6)
set_level_shifter_threshold (1.6)
set_load
set_logic_dc
set_logic_one
set_logic_zero
set_max_area
set_max_capacitance
set_max_dynamic_power (1.4)
set_max_fanout
set_max_leakage_power (1.4)
set_max_transition
set_min_capacitance
set_min_fanout
set_min_porosity (1.4)
set_operating_conditions
set_port_fanout_number
set_resistance
set_wire_load_min_block_size
set_wire_load_mode
set_wire_load_model
set_wire_load_selection_group
```

Like **write_script**, the **write_sdc** command writes out commands relative to the top of the design, regardless of the current instance. SDC files written by **write_sdc** must be read in from the top of the design.

For a complete guide to using SDC with Synopsys applications, see the *Using the Synopsys Design Constraints Format Application Note* which is available through SolvNET at <http://solvnet.synopsys.com>.

The usage of some of the supported commands is restricted when reading SDC. In some cases, some options are not allowed. The **write_sdc** command supports the restricted usage by restricting what is written. The following restrictions apply for SDC Version 1.3:

- For **set_driving_cell**, the **-min** and **-max** options are not supported.
- For **set_port_fanout_number**, the **-min** and **-max** options are not supported.

When the hierarchy has been partially flattened, embedded hierarchy separators can make names ambiguous. It is not clear which hierarchy separator characters are part of the name, and which are real separators. Beginning with SDC Version 1.2, hierarchical names can be made non-ambiguous using the **set_hierarchy_separator** SDC command and/or the **-hsc** option available on the **get_cells**, **get_lib_cells**, **get_lib_pins**, **get_nets**, and **get_pins** SDC object access commands. By default, PrimeTime and Design Compiler write an SDC file using these features to create non-ambiguous names. It is considered best practice to write SDC files that contain names that are not ambiguous. However, if you are using a third-party application that does not fully support SDC 1.2 or later (that is, it does not support the non-ambiguous hierarchical names features of SDC), then you can suppress these features by setting **sdc_write_unambiguous_names** variable to **true**.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following command writes the SDC script to the file named *top.sdc*:

```
prompt> write_sdc top.sdc  
1
```

SEE ALSO

`read_sdc(2)`
`write_script(2)`
`sdw_write_unambiguous_names(3)`

write_sdf

Writes a Standard Delay Format (SDF) back-annotation file.

SYNTAX

```
string write_sdf  
  [-version sdf_version] [-significant_digits digits]  
  [-instance inst_name]  
  file_name
```

Data Types

<i>sdf_version</i>	string
<i>inst_name</i>	string
<i>file_name</i>	string

ARGUMENTS

-version *sdf_version*

Selects which SDF version to use. Supported SDF versions are 1.0 or 2.1. SDF 2.1 is the default.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are reported. Allowed values are from 0 through 13; the default is 3. Using this option overrides the value set by the variable **report_default_significant_digits**.

-instance *inst_name*

Specifies that the SDF is to be written only for the instance named *inst_name*.

file_name

Specifies the name of the SDF file to write.

DESCRIPTION

Writes leaf cell pin-to-pin timing information to a disk file. Timing information is written in SDF format using version v1.0 or v2.1. The timing file contains data associated with the netlist from which it is created.

Use **write_sdf** only when the instance names in the design agree with the naming conventions of the system to which the timing file is written.

Timing information can be written in multiples of ns, ps, and us. The time unit in the SDF file is the time unit specified in the technology library and is written in the timing file under 'timescale'. If there is no time unit in the library, the unit is assumed to be a multiple of nanoseconds.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example writes timing information for the design **MULT16** to a disk file called mult16.sdf, using SDF version 2.1.

```
prompt> write_sdf -version 2.1 mult16.sdf
```

SEE ALSO

- read_sdf(2)
- set_annotated_check(2)
- set_annotated_delay(2)
- remove_annotated_check(2)
- remove_annotated_delay(2)
- report_annotated_check(2)
- report_annotated_delay(2)
- report_default_significant_digits(3)

write_test

Formats the test patterns for the current design into one or more test vector files.

SYNTAX

```
status write_test
  [-output output_vector_file_name]
  [-capture_cycle
  -format stil | wgl_serial | verilog
  [-multiple_pattern_counter list]
  [-multiple_signature list]
  [-multiple_seed list]
  [-multiple_occ list]
```

Data Types

output_vector_file_name string

ARGUMENTS

-output *output_vector_file_name*

Specifies as output the name of the path and base file name for the test program files. By default, test program files are written in the current directory, using the current design name as the base file name.

-capture_cycle *binary_word*

Specifies the self-test capture clock pattern to use. See the "Autonomous Self-Test Operation" section for details.

The valid values are **01** or **10** for a single capture pulse, and **11** for two capture pulses. (The tool zero-pads the capture pattern to the **set_dft_clock_controller -cycles_per_clock** value, if needed.) The default is a single capture pulse. This option requires that OCC clock weights also be configured.

-format stil | wgl_serial | verilog

Selects the format for the output test program. Valid values for *test_program_format* are as follows:

- Specifying **stil** generates STIL patterns.
- Specifying **wgl_serial** generates patterns in serial WGL format.
- Specifying **verilog** generates a Verilog RTL testbench.

DESCRIPTION

The **write_test** command writes out a test pattern file for the current design.

Test patterns can be written for the following:

- Boundary-scan logic
- Autonomous self-test (DFTMAX LogicBIST self-test)

Boundary-Scan Test Patterns

For boundary-scan logic, the **write_test** command writes the test patterns generated for the current design by the **create_bsd_patterns** command into a series of test program files suitable for running on automated test equipment (ATE) or simulators.

The output test program is intended as input to manufacturing ATE. The program specifies the stimuli to be applied to a fabricated chip and specifies the responses that a fully working chip generates. If during the testing process, the chip does not generate the correct responses when the stimuli are applied, the chip is faulty and should be discarded.

By default, all test vectors are placed into a single test program file.

The organization of the test program, in terms of the precise boundary scan test sequence employed, is determined by the boundary scan test protocol for the design inferred by the **create_bsd_patterns** command.

By default, the values used for the 4 timing options are those specified in the test protocol file for the design. In the protocol inferred by the **create_bsd_patterns** command, the default timing values are controlled by the environment variables: **test_default_period**, **test_bsd_default_strobe**, **test_bsd_default_delay**, and **test_bsd_default_bidir_delay**. In addition, the timing of any clock signals can again be specified in a test protocol file or can be specified directly by using the **set_dft_signal** command.

Autonomous Self-Test Operation

For designs with LogicBIST self-test, the **write_test** command writes patterns that perform autonomous self-test. The patterns load the signature value into the PRPG, run autonomous self-test to completion, then measure the resulting signature value against the expected value.

The self-test logic uses the following values to control its operation:

- Seed value
- Signature value
- Pattern counter
- Shift counter (number of shift cycles in each pattern)
- Capture clock mask

By default, the testbench takes no special action for these values. Constant-driven values are driven according to their value. Port-driven values require that the desired values be forced externally using simulator commands or options.

However, if you have implemented self-test logic (in the current design or within a core) that is controlled by DFT-inserted IEEE 1500 logic, the **write_test** command allows you to specify these values, and the testbench loads or compares the values through the IEEE 1500 interface as needed.

Autonomous self-test patterns can be written only in the STIL format.

EXAMPLES

The following example writes out a test program for a design named *HUNSLEY* using the STIL format:

```
prompt> current_design HUNSLEY
prompt> write_test -format stil
Getting test program from design.
Writing test patterns to file HUNSLEY.stil.
1
```

The following example creates a Verilog RTL testbench:

```
prompt> write_test -format verilog
Getting test program from design.
Writing native Verilog test bench to file HUNSLEY_verilog_test.v.
1
```

The following example writes a self-test testbench for a LogicBIST self-test design:

```
prompt> write_test -format stil -output selftest \
-seed 101110000000111111 \
-signature 011011000100011110 \
-shift_counter 101000 \
-pattern_counter 1110 \
-capture_cycle 11
...Getting test program from design.
...Writing test patterns to file selftest.stil.
1
```

SEE ALSO

- check_bsd(2)
- create_bsd_patterns(2)
- current_design(2)
- insert_dft(2)
- set_dft_signal(2)
- search_path(3)
- test_bsd_default_bidir_delay(3)
- test_bsd_default_delay(3)
- test_bsd_default_strobe(3)
- test_default_period(3)

write_test_model

Writes a test model file.

SYNTAX

```
int write_test_model  
  [-format ctl | ddc]  
  [-names verilog | verilog_single_bit]  
  [-output model_file]  
  [-inclusive]  
  [-design design_name]
```

Data Types

```
model_file    string  
design_name   string
```

ARGUMENTS

-format *ctl* | *ddc*

Specifies the format of the test model file. When **ddc** is specified, the CTL test model information and an interface-only representation of the design are written out to a file. When **ctl** is specified, an ASCII CTL test model file is written out to a file. The default format is **ddc**.

-names *verilog* | *verilog_single_bit*

Specifies the form of the names used in the ASCII CTL model. By default, names are not changed from internal representation. Names can be modified as Verilog names or as Verilog names compatible with the usage of the **verilogout_single_bit** environment variable. In all cases the internal representation is not changed.

-output *model_file*

Specifies the name of the output test model file. By default, the output file name is *design_name.format*.

-inclusive

Expands the include statement when writing out an ASCII CTL file. By default, the included contents are written out to separate files.

-design *design_name*

Specifies the name of the design for which the test model is written. The default is the current design.

DESCRIPTION

The **write_test_model** command writes a test model file to disk.

When **-format ddc** is specified, a test model for the specified design is written out as a .ddc file. This file contains test model information, as well as an interface-only representation of the design. This file can be used to satisfy link requirements for instantiations of that design. The **write_test_model** command does not write out any netlist information for the specified design. If you need both the test model information as well as the full netlist, you should use the **write -format ddc** command instead.

When **-format ctl** is specified, a test model is written to an ASCII CTL file. The ASCII CTL format is primarily used for exporting test models outside of a Synopsys environment. The **write_test_model** command does not modify the version of the design that is in memory.

The **write_test_model** command is insensitive to the **test_stil_netlist_format** application variable.

EXAMPLES

The following example writes out a test model file in .ddc format:

```
prompt> write_test_model -format ddc -output ALARM_SM_2.ddc
Writing test model file
'/home/felixng/DFTC/dft_tutorial/ctl/ ALARM_SM_2.ddc'...
1
```

The following example writes out a test model file in CTL format:

```
prompt> write_test_model -format ctl -output ./ctl/ALARM_SM_2.ctl
Writing test model file
'/home/felixng/DFTC/dft_tutorial/ctl/ALARM_SM_2.ctl'...
1
```

SEE ALSO

list_test_models(2)
read_test_model(2)
remove_test_model(2)
report_test_model(2)
write(2)

write_test_protocol

Writes a STIL test protocol file.

SYNTAX

```
int write_test_protocol
  [-design design_name]
  [-output file_name]
  [-test_mode mode_name]
  [-instruction instruction_name]
  [-names format_name]
  [-disable_codecs decompressor_list]
```

Data Types

<i>design_name</i>	string
<i>file_name</i>	string
<i>mode_name</i>	string
<i>format_name</i>	string
<i>decompressor_list</i>	list

ARGUMENTS

-design *design_name*

Specifies the design name for which the STIL test protocol is written. The default is the current design.

-output *file_name*

Specifies the name of the text file to which the STIL test protocol is written. If a *file_name* is not specified, the file is named *current_design_name.spf*.

-test_mode *mode_name*

Specifies the test mode from which the STIL protocol is generated.

-instruction *instruction_name*

Specifies that a STIL protocol file should be written for the specified JTAG instruction implemented by BSD Compiler. This option can be used after the **insert_dft** or **check_bsd** commands have been run.

-names *format_name*

Specifies the form of the names used in the STIL protocol. Valid values for *format_name* are as follows:

- The value of **dc_shell** instructs DFT Compiler to write a STIL protocol where the design objects are not changed.

- The value of **verilog** instructs DFT Compiler to write a STIL protocol where the names of the design objects follow the naming rules of the Verilog format. Vectored ports are allowed.
- The value of **verilog_single_bit** instructs DFT Compiler to write a STIL protocol where the names of the design objects follow the naming rules of the Verilog format. The STIL protocol file will not contain vectored ports, but it will have single bit ports.

Names can be unchanged from internal representation (the default). They can be modified as Verilog names or as Verilog names compatible with the usage of the **verilogout_single_bit** environment variable. In all cases, the internal representation is not changed. This option takes effect only in conjunction with **-test_mode** options, when Hierarchical Scan Synthesis is used. In all other cases, the form of the names is determined by the setting of the **test_stil_netlist_format** variable.

-disable_codecs decompressor_list

Specifies the codecs to be disabled in the test protocol. The default is not to disable any codecs.

Codecs are referenced by decompressor name. You can disable any set of codecs, provided at least one codec is active in the design across all DFT partitions.

If you are using IEEE 1500 test-mode control, this command writes a test protocol that automatically configures the TMCDR codec selection bits as needed.

If you are not using IEEE 1500 test-mode control, this command assumes that you are modifying the test protocol and/or netlist to disable the specified codecs, and it issues a TEST-1473 information message to remind you of this.

This option can only be used for codecs that have codec controls. To insert codecs with codec controls, use the **-shared_codec_controls** option of the **set_scan_compression_configuration** command.

DESCRIPTION

The **write_test_protocol** command outputs the STIL test protocol from memory to disk in text format. This allows you to customize the test protocol.

The scan test protocol formally defines the process by which a design is tested, so the sequence of scan-in vectors, parallel vectors, and scan-out vectors is performed for each test pattern. The protocol defined for a design is used to drive scan test and design rule checking.

If a *file_name* is not specified with the **-output** option, the file is named *current_design_name.spf*.

If the **-test_mode** option is used, the protocol for the specified test mode is written out.

when an OCC (On-Chip Clocking) controller block is present in the design, the protocols for both the "PLL clocks bypassed" and "PLL clocks non-bypassed" modes will be included in the same protocol file.

If the **-instruction** option is used, the protocol is generated for the specified JTAG instruction implemented by BSD Compiler. The instruction name corresponds to the name given to the **set_bsd_instruction** command. This option can be used after one of the following commands has been successfully run:

- **insert_dft**, with BSD synthesis enabled
- **check_bsd**

EXAMPLES

The following example writes a test protocol to the **newUPC1.spf** file:

```
prompt> write_test_protocol -output newUPC1.spf  
Writing test protocol file 'newUPC1.spf'...
```

The following example writes a test protocol with two codecs disabled:

```
prompt> write_test_protocol -test_mode MY_COMP \  
       -output MY_COMP.spf \  
       -disable_codecs { \  
           {sub1:sub_1_U_decompressor_ScanCompression_mode \  
           sub2:sub_2_U_decompressor_ScanCompression_mode}  
Writing test protocol file 'MY_COMP.spf'...
```

The following example writes a test protocol for the **UPC1** design to the default file named **UPC1.spf**:

```
prompt> write_test_protocol  
Writing test protocol file 'UPC1.spf'...
```

The following example writes a test protocol for the scan-through-TAP (STT) JTAG instruction:

```
prompt> set_bsd_instruction STT -register STT_REG -code 1101  
...  
prompt> write_test_protocol -instruction STT -output BSD_STT.spf  
Writing test protocol file 'BSD_STT.spf'...
```

SEE ALSO

- `create_test_protocol(2)`
- `current_design(2)`
- `dft_drc(2)`
- `read_test_protocol(2)`
- `remove_test_protocol(2)`
- `insert_dft(2)`
- `check_bsd(2)`

write_timing_context

Writes Scenarios constraints and configuration for the current design

SYNTAX

```
status write_timing_context  
-output directory  
[-scenarios scenario_list]  
[-format icc2]  
[-nosplit]
```

ARGUMENTS

-scenarios *scenario_list*

Specifies a list of active and inactive Scenarios for which the timing context will be written for loading in IC Compiler II

-format *icc2*

Specifies for which target tool the timing contexts are to be written. Currently, only the icc2 format is supported.

-nosplit

This argument behaves the same as the -nosplit argument of the **write_script** command.

DESCRIPTION

This command writes a directory containing timing contexts (Scenarios) in a format that may be read into IC Compiler II in order to create and populate the same set of Scenarios as specified in IC Compiler. The output directory will contain a timing context constraint file for each of the specified (or all) Scenarios and scripts to create and populate the IC Compiler II modes, corners and scenarios. The directory structure of the constraint directory created by **write_timing_context** is the same as written by the IC Compiler II **write_script** command

```
top.tcl           - Mode, Corner and Scenario creation and population script.  
design.tcl         - Scenario-independant timing constraints  
scenario_Scen1.tcl - Scenario-specific constraints for the Scenario Scen1  
scenario_Scen2.tcl - Scenario-specific constraints for the Scenario Scen2
```

Multicorner-Multimode Support

By default, this command uses information from all scenarios

EXAMPLES

The following example uses the **write_timing_context** command to create a constraint directory suitable for use in IC Compiler II.

```
icc> get_scenarios -setup true -active true;  
Func::COLD Func::HOT
```

```
icc> write_timing_context -format icc2 -nosplit -verbose -output CONS4ICC2 -scenarios [get_scenarios -setup true -active true];
```

```
icc.2> open_block netlist;  
icc.2> source CONS4ICC2/top.tcl;  
icc.2> get_object_names [get_scenarios -filter setup&&active];  
Func::HOT Func::COLD
```

SEE ALSO

[write_script\(2\)](#)