4

Compile Flow

Learning Objectives

During this lab, you will study Design Library setup, Constraints, Multi Corner Multi Mode setup, compile_fusion flow and collect reports for design analyses.

After completing this lab, you should be able to:

- Work with blocks in the design library
- Design Library setup
- Setup the design constraints
- Create Multi Corner Multi Mode constraints
- Implement compile_fusion stages
- Collect reports

Lab Duration:
40 minutes

Lab Instructions

Lab work structure

Lab directory organized as follows:

```
FC_Labs/labs/lab4_compile_flow/
   scripts/
        compile_flow.tcl
   work/
        logs/
        Makefile
```

Move to work directory:

```
cd FC_Labs/labs/lab4_compile_flow/work
```

You can run this lab by using Makefile in work directory, for that use the following command in terminal:

```
make run
```

Or invoke the Fusion Compiler and source the **compile_flow.tcl** script:

```
fc_shell -gui -no_log -f ../scripts/compile_flow.tcl | tee
./logs/compile_flow.log
```

In case you want to rerun the lab, you can clean your working area by using the following command in terminal:

```
make clean
```

At the beginning of lab work you **must** source the following setup scripts in Fusion Compiler to correcly setup the flow:

```
#### Sourcing common setup script
source -echo ../../setup/fc_common_setup.tcl
#### Sourcing flow setup script
source -echo ../../setup/fc_flow_setup.tcl
```

compile_fusion Flow

The Fusion Compiler tool performs unified physical synthesis and supports complete RTL_to_GDS flow. Fusion Compiler combines traditional logic synthesis with floorplan creation, placement and optimization.

Unified physical synthesis is done by implementing **compile_fusion** command:

```
compile_fusion
```

In this lab work scandef file is not used, so before starting the synthesis, the following application option must be set to allow the tool to perform placement and optimization without scandef file.

```
set_app_options -name \
    place.coarse.continue_on_missing_scandef -value true
```

To check the design before synthesis, use the following command:

```
compile_fusion -check_only
```

The **compile_fusion** command consists of the following 7 stages, which you can implement separately:

1. **initial_map:** Design mapping and area optimization.

```
compile_fusion -to initial_map
```

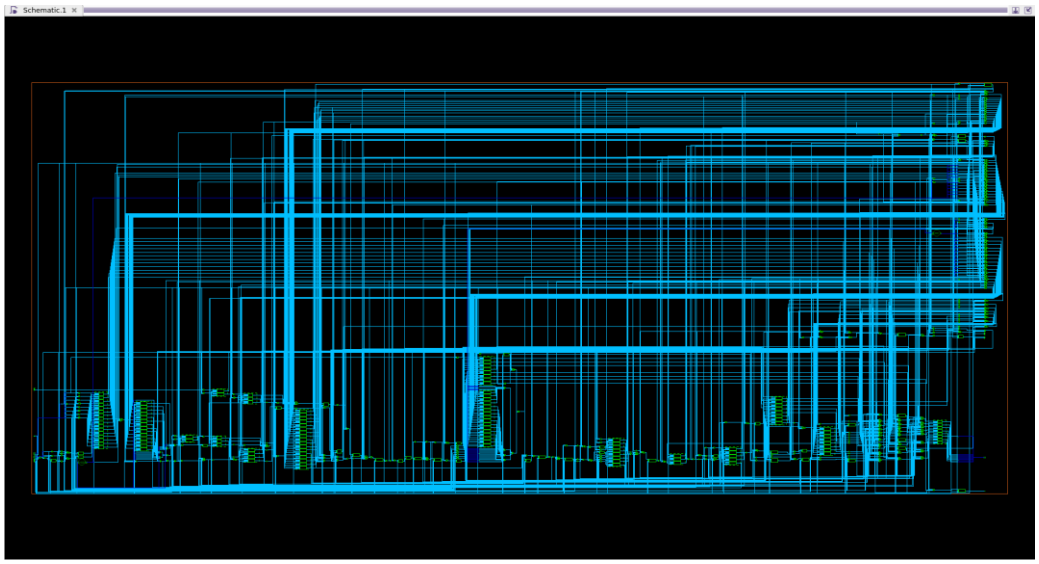

Fig. Schematic View after initial_map stage

Now, the logic circuit is **technology-dependent**, as it consists of standard cells from the provided technology library.

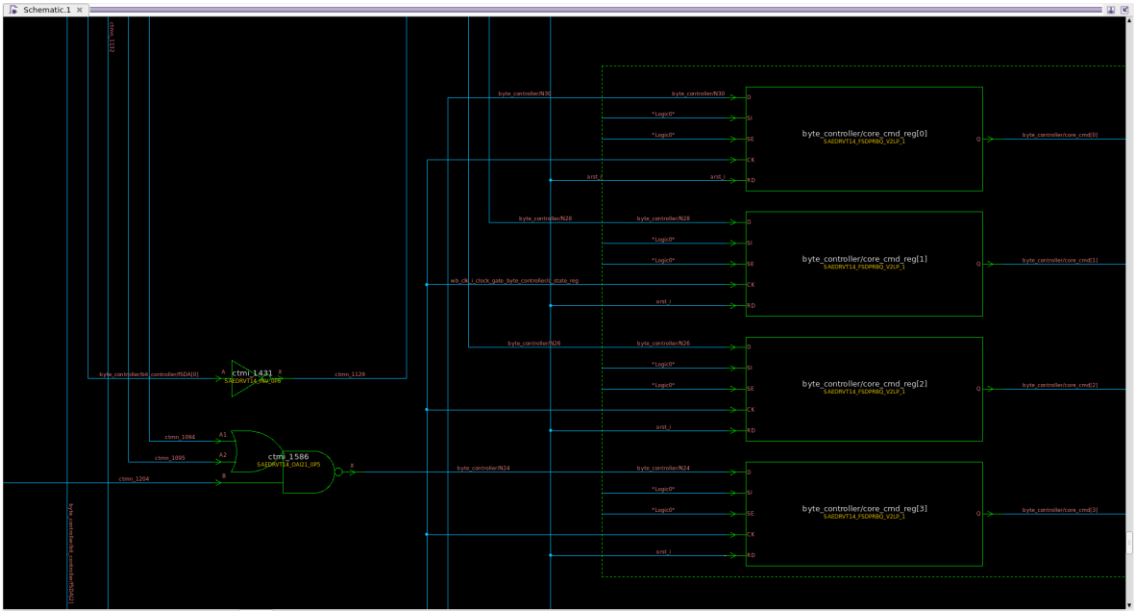

Fig. Standard cells from the SAED14 RVT library in the Schematic View

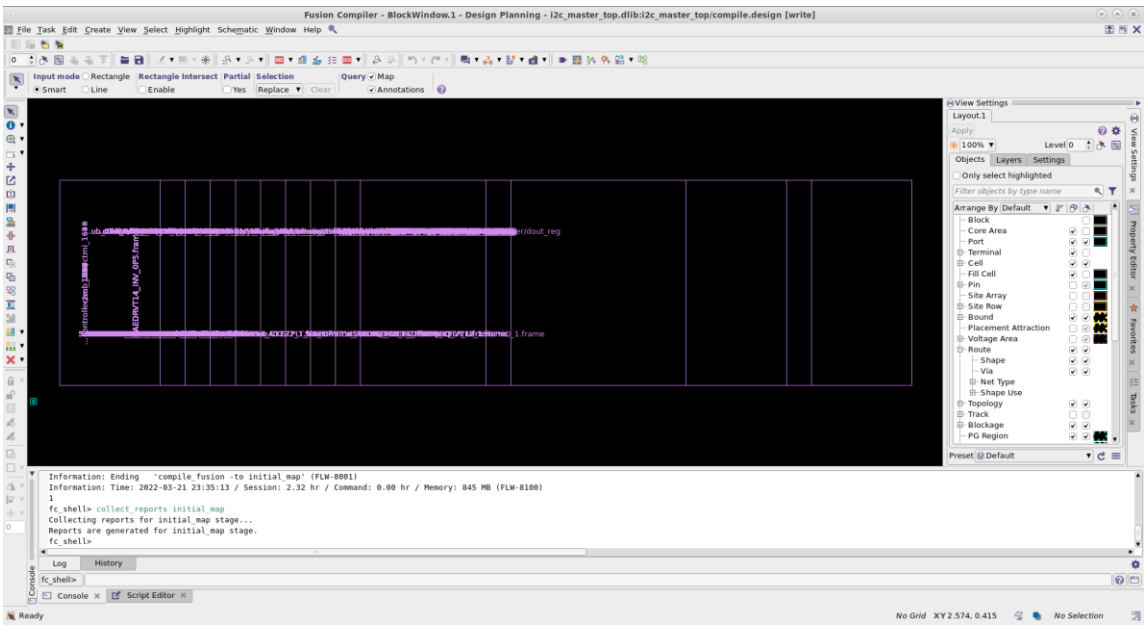

Fig. Layout View after initial_map stage

Used cells are visible and collected in one place.

2. **logic_opto:** Logic-based delay optimization. ***Automatic floorplan creation*.**

```
compile_fusion -from logic_opto -to logic_opto
```

An automatic floorplan is created and cells are initially placed. Please note that this floorplan can not be used for real design implementation. It must be tuned. Floorplan creation will be described in the next lab work.

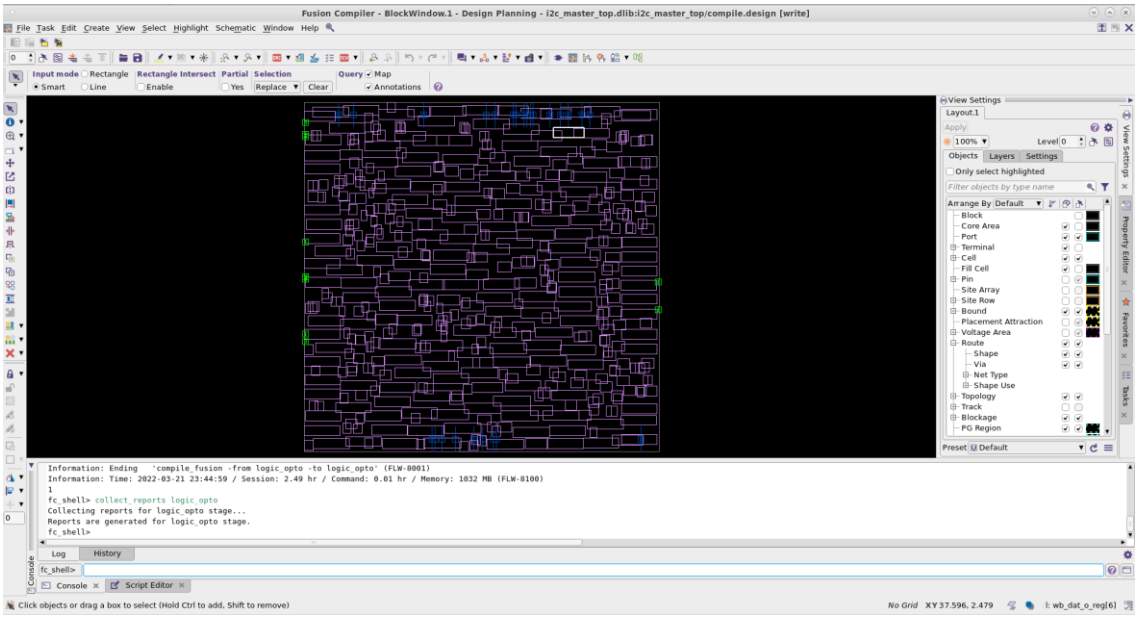

Fig. Layout View after logic_opto stage

3. **initial_place: C**oarse placement.

```
compile_fusion -from initial_place -to initial_place
```

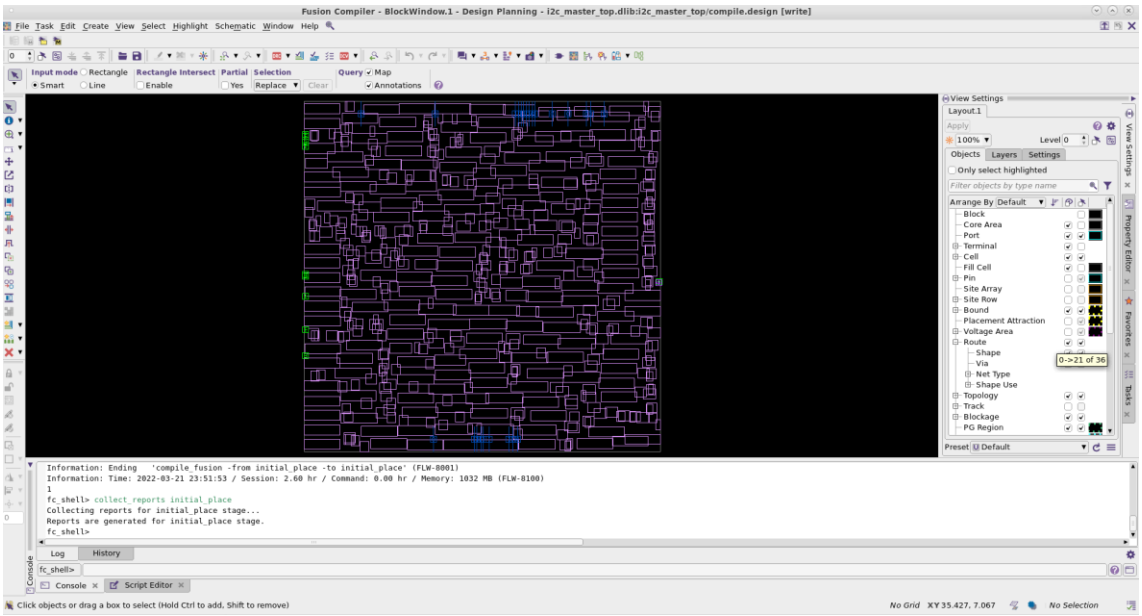

Fig. Layout View after initial_place stage

4. **initial_drc:** Buffer tree creation for high-fanout nets and design rule violation fixing.

```
compile_fusion -from initial_drc -to initial_drc
```

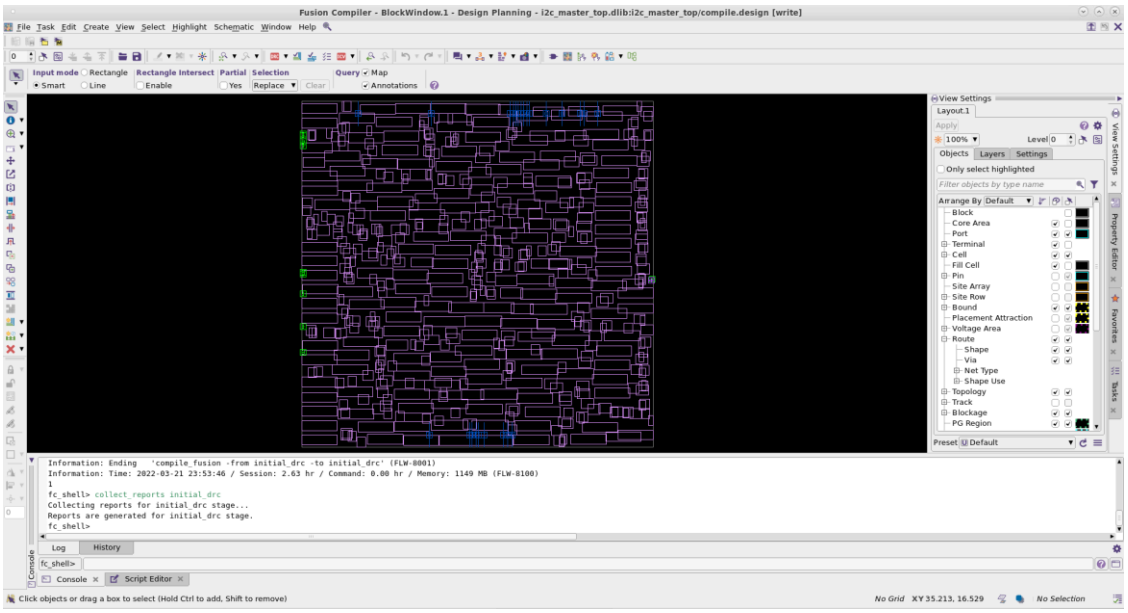

Fig. Layout View after initial_drc stage

5. **initial_opto:** Incremental placement and optimization.

```
compile_fusion -from initial_opto -to initial_opto
```

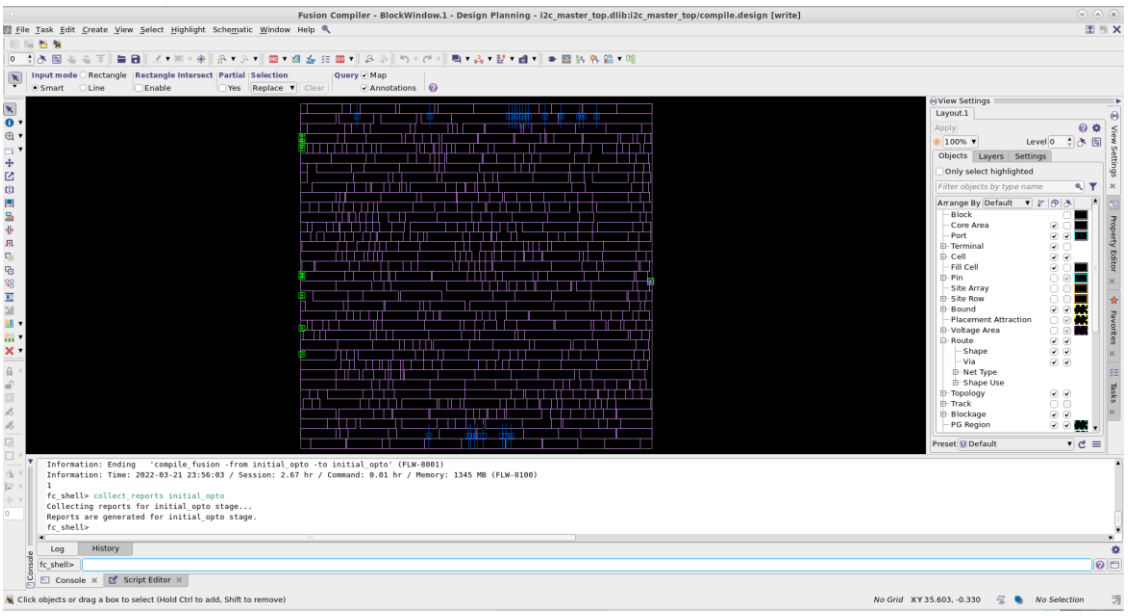

Fig. Layout View after initial_opto stage

The design is now legally placed.

6. **final_place:** Final placement to improve timing and congestion.

```
compile_fusion -from final_place -to final_place
```

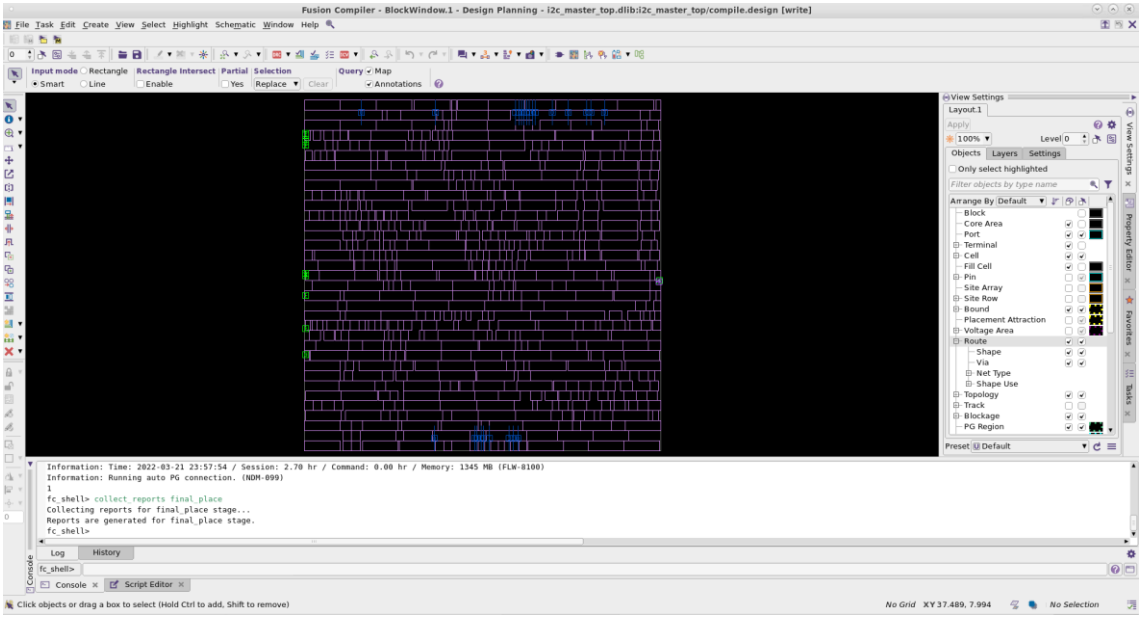

Fig. Layout View after final_place stage

7. **final_opto:** Final optimization and legalization.

```
compile_fusion -from final_opto -to final_opto
```

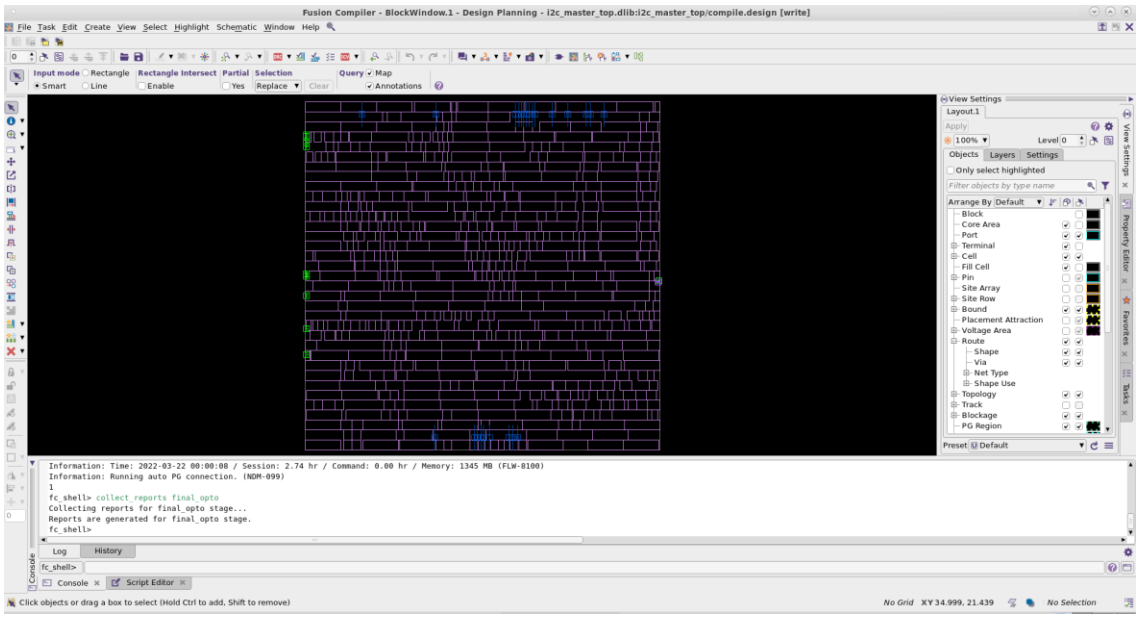

Fig. Layout View after final_opto stage

Each stage of the compile_fusion command modifies and optimizes the design, so you can collect the reports after each stage and compare them.

To check the legality of the placement, use the following command:

```
check_legality
```

Collecting the reports

After each stage reports are generated for review. The reports are supporting MCMM setup, so they can be collected for different scenarios for comparison and analysis.

1. To report design's **power** consumption for all scenarios, use the following command:

```
report_power -scenarios [all_scenarios]
```

The report will show the leakage, internal, switching and total power values for the different cell groups and scenarios.

```
Mode: FUNC
Corner: Typical
Scenario: FUNC_Typical
Voltage: 0.80
Temperature: 25.00

Voltage Unit          : 1V
Capacitance Unit      : 1fF
Time Unit             : 1ns
Temperature Unit      : 1C
Dynamic Power Unit    : 1pW
Leakage Power Unit    : 1pW

Switched supply net power scaling:
scaling for leakage power

Supply nets:
VDD (power) probability 1.00 (default)
VSS (ground) probability 1.00 (default)

  Cell Internal Power    = 7.84e+07 pW ( 82.3%)
  Net Switching Power    = 1.68e+07 pW ( 17.7%)
Total Dynamic Power      = 9.53e+07 pW (100.0%)

Cell Leakage Power       = 1.02e+05 pW


  Attributes
  ----------
      u  -  User defined power group

Power Group          Internal Power        Switching Power          Leakage Power            Total Power    (   %  )    Attrs
--------------------------------------------------------------------------------------------------------------------------------
io_pad                    0.00e+00              0.00e+00              0.00e+00              0.00e+00    (  0.0%)
memory                    0.00e+00              0.00e+00              0.00e+00              0.00e+00    (  0.0%)
black_box                 0.00e+00              0.00e+00              0.00e+00              0.00e+00    (  0.0%)
clock_network             7.49e+07              1.13e+07              3.71e+03              8.62e+07    ( 90.4%)
register                  2.67e+06              9.15e+05              6.31e+04              3.65e+06    (  3.8%)
sequential                0.00e+00              0.00e+00              0.00e+00              0.00e+00    (  0.0%)
combinational             8.87e+05              4.56e+06              3.53e+04              5.48e+06    (  5.7%)
--------------------------------------------------------------------------------------------------------------------------------
Total                     7.84e+07 pW           1.68e+07 pW           1.02e+05 pW           9.54e+07 pW
1
```

Fig. Example of power report

2. To report design's **area**, use the following command:

```
report_area
```

```
Number of ports:                          35
Number of nets:                          541
Number of cells:                         520
Number of combinational cells:           356
Number of sequential cells:              164
Number of macros/black boxes:              0
Number of buf/inv:                        56
Number of references:                     43

Combinational area:                   115.35
Buf/Inv area:                          10.39
Noncombinational area:                205.53
Macro/Black Box area:                   0.00

Total cell area:                      320.88
1
```

Fig. Example of area report

3. To report design's critical path for **setup** time for all scenarios, use the following command:

```
report_timing -scenarios [all_scenarios] -delay_type max
```

```
Startpoint: byte_controller/bit_controller/cnt_reg[2] (rising edge-triggered flip-flop clocked by wb_clk_i)
Endpoint: byte_controller/bit_controller/cnt_reg[15] (rising edge-triggered flip-flop clocked by wb_clk_i)
Mode: FUNC
Corner: Slow
Scenario: FUNC_Slow
Path Group: wb_clk_i
Path Type: max

Point                                                   Incr      Path
-----------------------------------------------------------------------
clock wb_clk_i (rise edge)                              0.00      0.00
clock network delay (ideal)                             0.00      0.00

byte_controller/bit_controller/cnt_reg[2]/CK (SAEDRVT14_FSDPRBQ_V2LP_1)
                                                        0.00      0.00 r
byte_controller/bit_controller/cnt_reg[2]/Q (SAEDRVT14_FSDPRBQ_V2LP_1)
                                                        0.10      0.10 f
ctmi_1451/X (SAEDRVT14_OR4_1)                           0.03      0.13 f
ctmi_177/X (SAEDRVT14_OR2_MM_0P5)                       0.02      0.15 f
byte_controller/bit_controller/sub_228/ctmi_1645/X (SAEDRVT14_NR2_1)
                                                        0.01      0.16 r
ctmTdsLR_1_351/X (SAEDRVT14_ND2B_U_0P5)                 0.02      0.18 f
byte_controller/bit_controller/sub_228/ctmi_1643/X (SAEDRVT14_NR2_1)
                                                        0.02      0.19 r
ctmTdsLR_1_349/X (SAEDRVT14_ND2B_U_0P5)                 0.02      0.21 f
byte_controller/bit_controller/sub_228/ctmi_1641/X (SAEDRVT14_NR2_1)
                                                        0.02      0.23 r
ctmTdsLR_1_347/X (SAEDRVT14_ND2B_U_0P5)                 0.02      0.25 f
byte_controller/bit_controller/sub_228/ctmi_1639/X (SAEDRVT14_NR2_1)
                                                        0.02      0.26 r
ctmTdsLR_1_345/X (SAEDRVT14_ND2B_U_0P5)                 0.02      0.28 f
byte_controller/bit_controller/sub_228/ctmi_1637/X (SAEDRVT14_NR2_1)
                                                        0.01      0.29 r
phfnr_buf_320/X (SAEDRVT14_INV_0P5)                     0.01      0.31 f
byte_controller/bit_controller/sub_228/ctmi_1666/X (SAEDRVT14_NR2_1)
                                                        0.01      0.31 r
byte_controller/bit_controller/sub_228/ctmi_1665/X (SAEDRVT14_EO2_V1_0P75)
                                                        0.02      0.33 r
ctmi_1569/X (SAEDRVT14_AO32_U_0P5)                      0.02      0.36 r
byte_controller/bit_controller/cnt_reg[15]/D (SAEDRVT14_FSDPRBQ_V2LP_1)
                                                        0.00      0.36 r
data arrival time                                                 0.36

clock wb_clk_i (rise edge)                              2.00      2.00
clock network delay (ideal)                             0.00      2.00
byte_controller/bit_controller/cnt_reg[15]/CK (SAEDRVT14_FSDPRBQ_V2LP_1)
                                                        0.00      2.00 r
clock uncertainty                                      -0.30      1.70
library setup time                                      0.02      1.72
data required time                                                1.72
-----------------------------------------------------------------------
data required time                                                1.72
data arrival time                                                -0.36
-----------------------------------------------------------------------
slack (MET)                                                       1.36


1
```

Fig. Example of timing report (setup time)

4. To report design's critical path for **hold** time for all scenarios, use the following command:

```
report_timing -scenarios [all_scenarios] -delay_type min
```

```
Startpoint: wb_ack_o_reg (rising edge-triggered flip-flop clocked by wb_clk_i)
Endpoint: wb_ack_o_reg (rising edge-triggered flip-flop clocked by wb_clk_i)
Mode: FUNC
Corner: Fast
Scenario: FUNC_Fast
Path Group: wb_clk_i
Path Type: min

Point                                             Incr      Path
-----------------------------------------------------------------
clock wb_clk_i (rise edge)                        0.00      0.00
clock network delay (ideal)                       0.00      0.00

wb_ack_o_reg/CK (SAEDRVT14_FSDPQ_V2LP_1)          0.00      0.00 r
wb_ack_o_reg/Q (SAEDRVT14_FSDPQ_V2LP_1)           0.03      0.03 f
ctmi_1562/X (SAEDRVT14_NR2_1)                     0.00      0.04 r
wb_ack_o_reg/D (SAEDRVT14_FSDPQ_V2LP_1)           0.00      0.04 r
data arrival time                                           0.04

clock wb_clk_i (rise edge)                        0.00      0.00
clock network delay (ideal)                       0.00      0.00
wb_ack_o_reg/CK (SAEDRVT14_FSDPQ_V2LP_1)          0.00      0.00 r
library hold time                                 0.02      0.02
data required time                                          0.02
-----------------------------------------------------------------
data required time                                          0.02
data arrival time                                          -0.04
-----------------------------------------------------------------
slack (MET)                                                 0.02


1
```

Fig. Example of timing report (hold time)

For these labs, a utility is created for collecting various reports. This utility is demonstrated below:

```
proc collect_reports {stage_name} {
        echo "Collecting reports for ${stage_name} stage..."
        redirect -file ../../reports/${stage_name}_report_power.rpt {report_power -scenarios [all_scenarios]}
        redirect -file ../../reports/${stage_name}_report_timing_setup.rpt {report_timing -scenarios [all_scenarios] -delay_type max}
        redirect -file ../../reports/${stage_name}_report_timing_hold.rpt  {report_timing -scenarios [all_scenarios] -delay_type min}
        redirect -file ../../reports/${stage_name}_report_area.rpt {report_area}
        redirect -file ../../reports/${stage_name}_report_qor.rpt  {report_qor}
        redirect -file ../../reports/${stage_name}_report_design.rpt  {report_design}
        echo "Reports are generated for ${stage_name} stage."
}
```

Fig. collect_reports utility

Above script is **FC_Labs/labs/setup/utilities.tcl**. It needs to be sourced in the current session for usage. Usage in fc_shell:

```
collect_reports ${STAGE_NAME}
```

After implementation the reports will be generated under the following path **FC_Labs/labs/reports** with following naming **${STAGE_NAME}_${REPORT_NAME}.rpt**

To finish the laboratory work, save and close the current block and library:

```
save_block
save_lib
close_block
close_lib
```