Project Description

Netlist to GDSII implementation of 32-bit RISC processor on Synopsys ICC - 28nm Process

Tool used:

Design Compiler for *Logic Synthesis*, **Formality for** *Formal Verification*, **IC Compiler for** *automatic P&R,* **StarRC for** *parasitic extraction*, **Primetime for** *static timing analysis and timing ECO generation.*

Design description:

No. of leaf cells = ~52K, No. of Macros or black boxes = 40

Chip area = ~583,136 um^2

Timing Corner – Signle (Functional Mode SDC)

Voltage Domain – Single (0.95V)

Phase:1 **Logic Synthesis** .v | .db | SDC

This project is a VLSI physical design implementation of 32-bit RISC processor for Synopsys 28/32nm CMOS. The project flow was developed for Synopsys tools and Mentor Caliber. During the starting phase of the project, the RTL design files was used for synthesis on Design Compiler. The synthesis was done using TCL script created for DC in which link library, target library and synthesis commands were defined. Some commands were also included in the script for timing, area, power and qor report generation. Another part of the synthesis was to put constraint for area and timing which later converted in .sdc file for the backend flow. The gate level netlist, reports (timing, area and power) and .sdc files were generated from DC.

Phase: 2 **Formal Verification** | Reference & Implemented design

In the next phase of the flow was the logic equivalence check. This was a quick step where golden netlist (RTL) was compared on the basis of Boolean equivalence with the implemented gate level netlist. The step was executed in three steps: read, match and verify. As the compared logically, the gate level netlist was ready for physical design implementation.

Phase: 3 **Physical Design** SDC | *_mapped.v | TLUplus (min, max) | map file

Physical design on Synopsys ICC was the most important and time consuming phase for this project. The goal was to successfully complete implementation within the limit of timing and design rule constraints. The physical design was done in following steps:

- Floorplaning and Placement of Macros
- Placement of Standard cells
- Clock Tree Synthesis
- Routing

Design was imported after setting the libraries (minTLUplus, maxTLUplus, tech2itf_map, link, target, physical) and the design variables (input_verilog, input_sdc, topcell). This script also created the mw database inside the design workspace and linked the physical library. Next, floorplan was created with core utilization of 70%, aspect ratio 1 and core to die difference of 10 um. During the floorplan step, ICC showed all the macros and standard cells outside the die area.

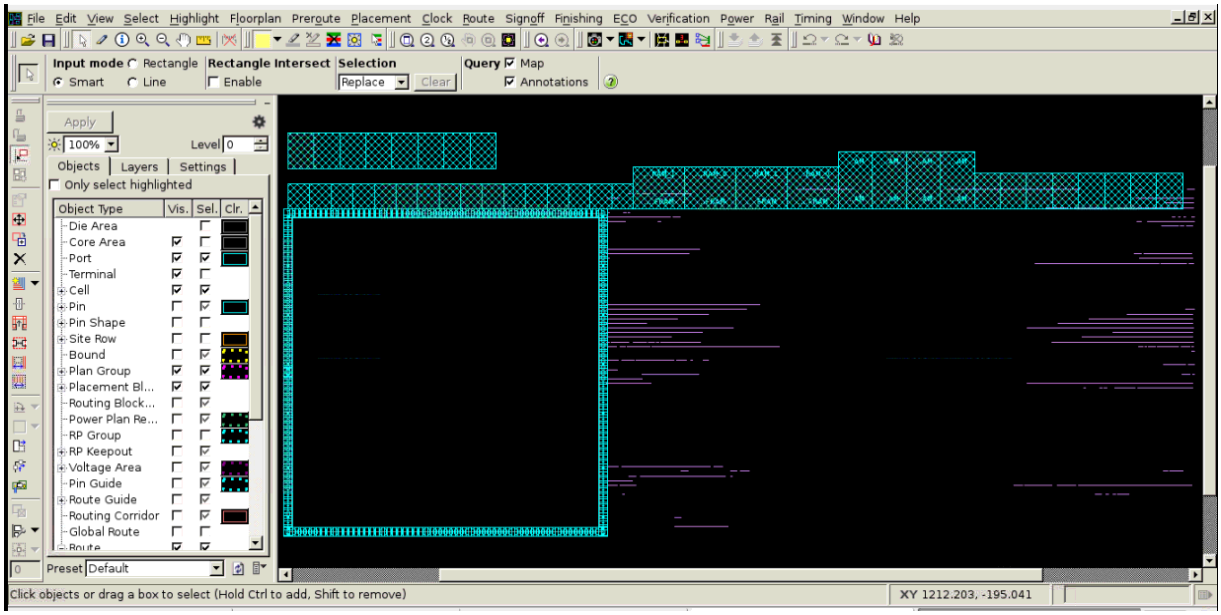

Figure:1 Floorplan Creation

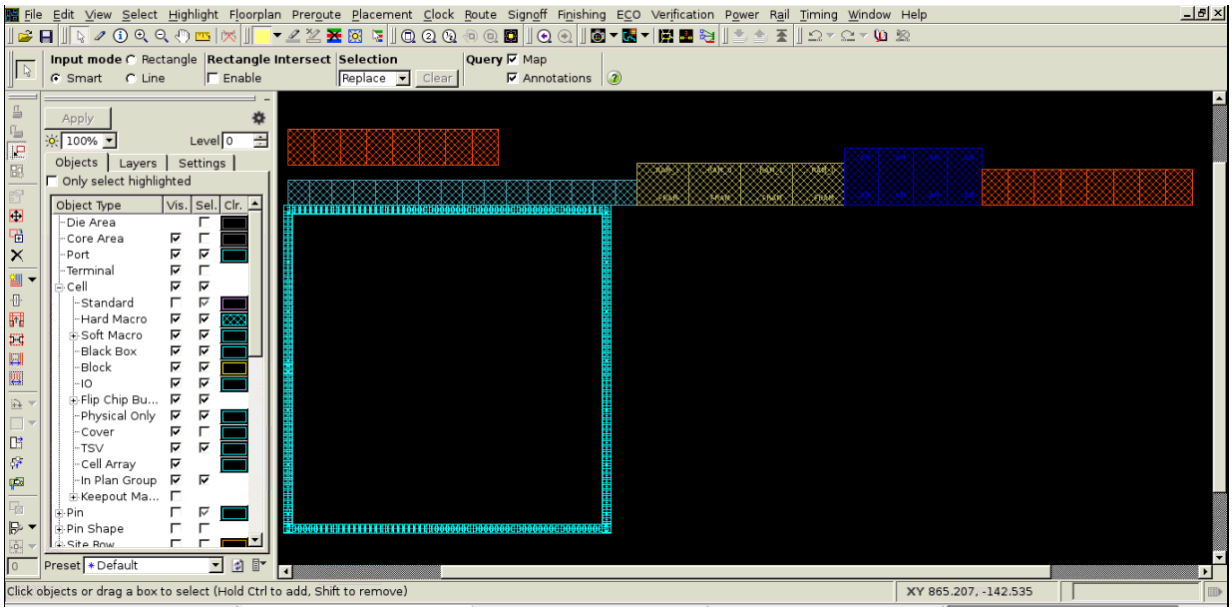

Figure:2 Macro Coloring by Hierarchy

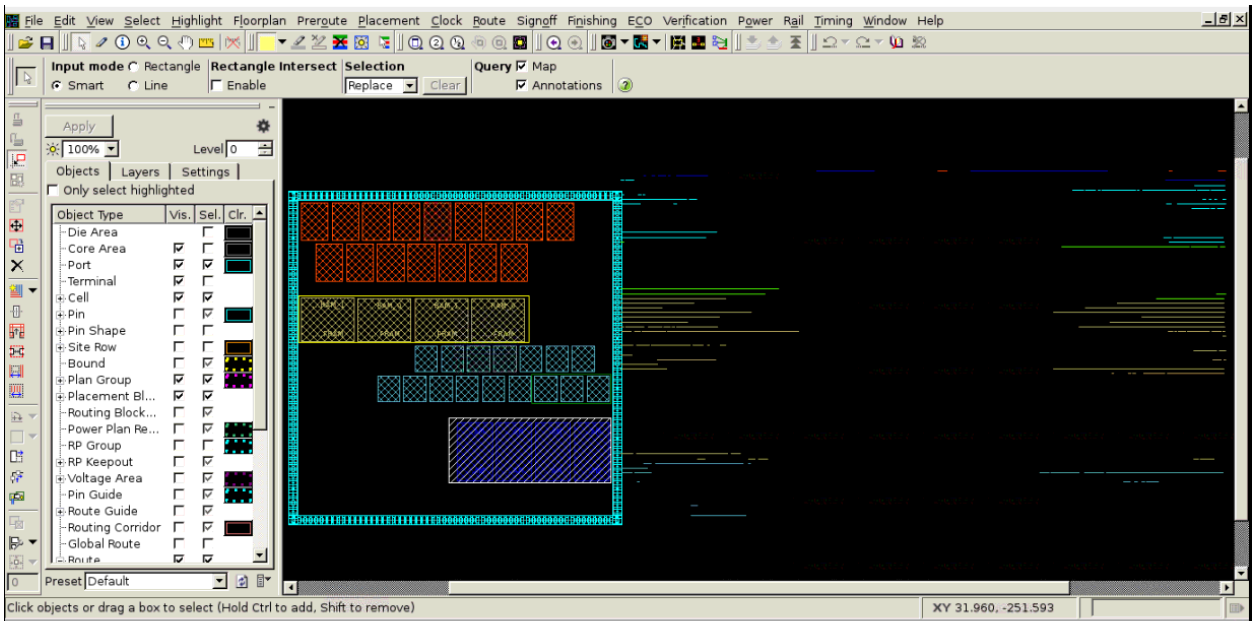

Figure:3 Macro Placement with blockages

Description of Macros:

1. Red Color: Low Power SRAM 2RW 64x8 (Context Memory) with 57 pins each
2. Blue Color: Low Power SRAM 2RW 128x16 (Register Files) with 91 pins each
3. Yellow Color: Low Power SRAM 2RW 64x32 (Read/Write FIFO) with 153 pins each

4. Green Color: Low Power SRAM 2RW 34x4 (Read/Write FIFO) with 39 pins each

Macro placement, alignment and distribution were done manually with the offset value of 10 micrometer between all the macros. Checking the cross connection using Fly-line analysis was the important part of the manual macro placement. The TCL scripts was used insert physical cells and later input-output buffer cells. The design is shown below with power routing.

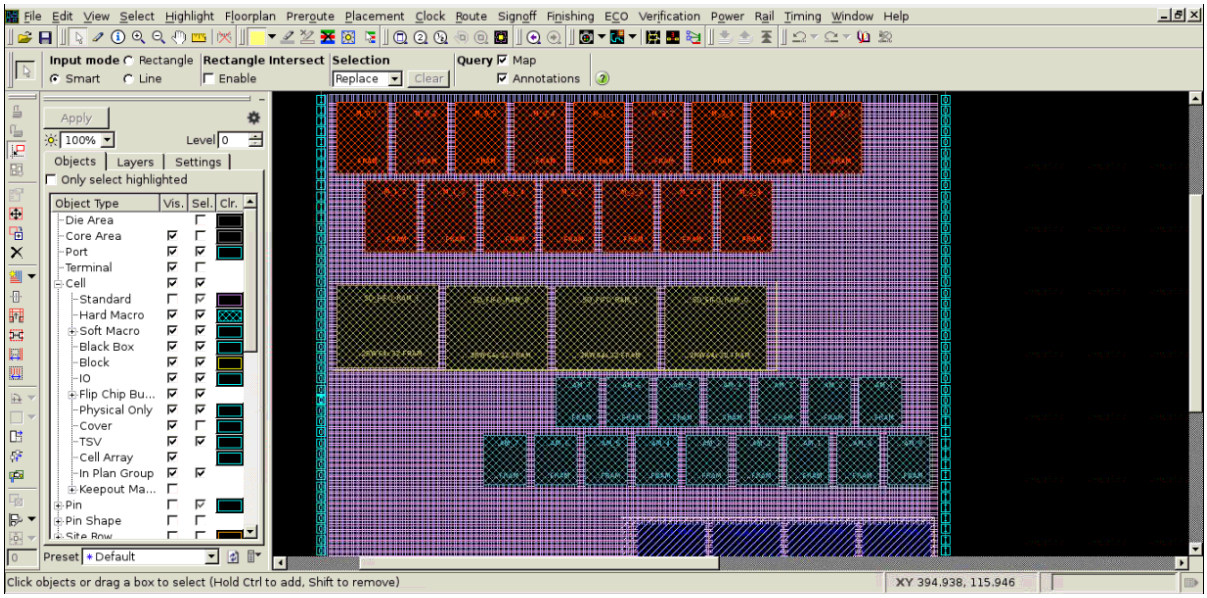

Figure:4 Power-Ground Net generation

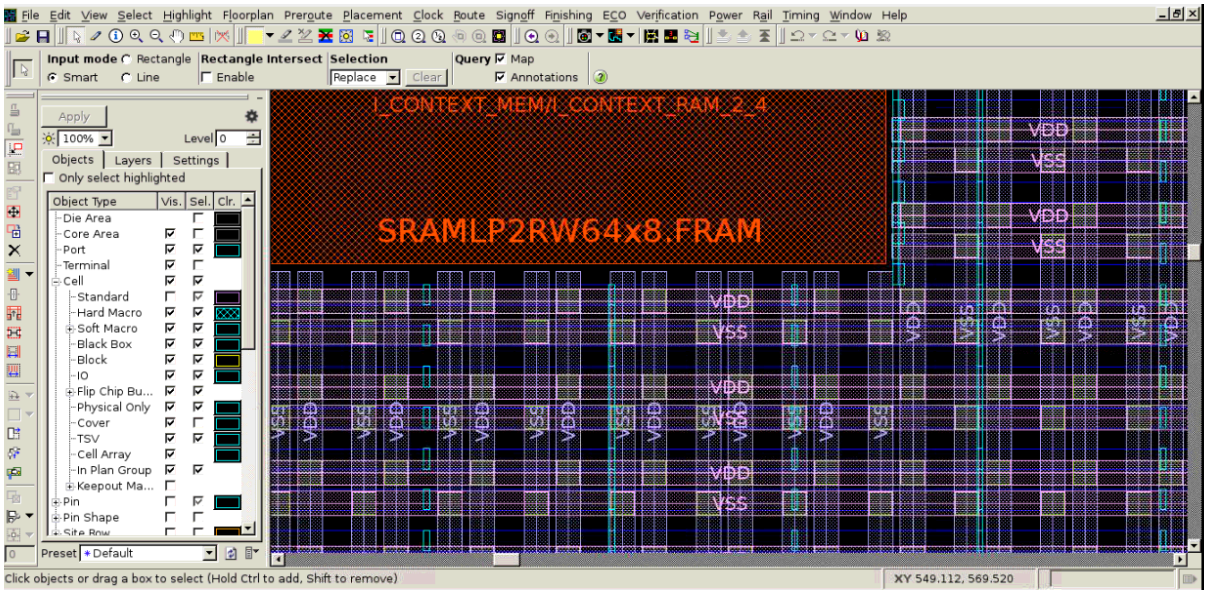

Figure:5 Power- Ground Net view with tap cells

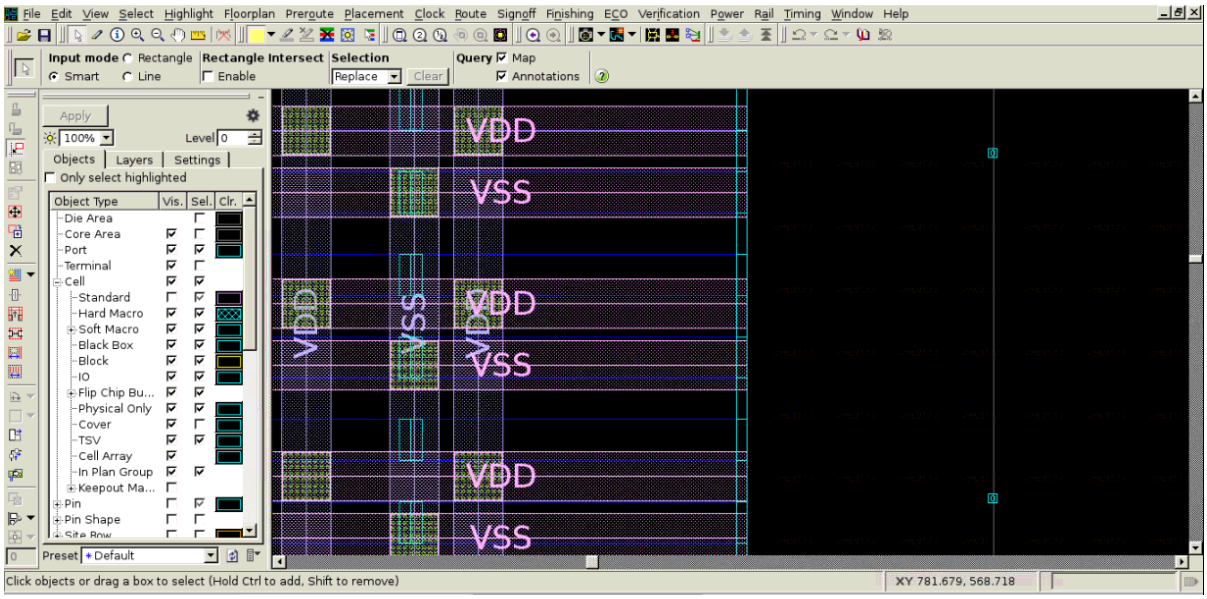

Figure:6 End cap cells

The standard cell placement was done after power routing using script. After this step all standard cells were placed inside the core area. First coarse placement (global placement) was executed which place all the standard cell inside the core are but to make sure there is no legalization issue, legalize placement was executed after checking overlaps of standard cells. Finally, place_opt command was used to take care of timing, congestion, legalization, AHFS (automatic high fanout synthesis) and optimization. The design was checked for quality by reporting congestion, power, legality, timing and qor to proceed for next stage.

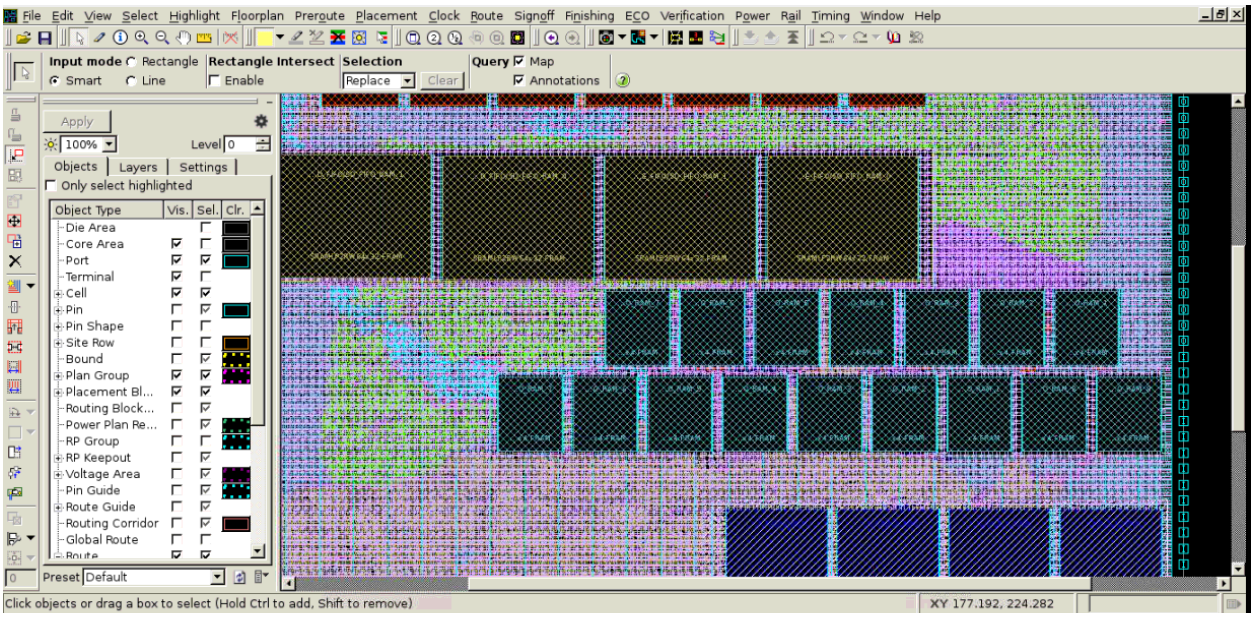

Figure:7 Standard cells placement

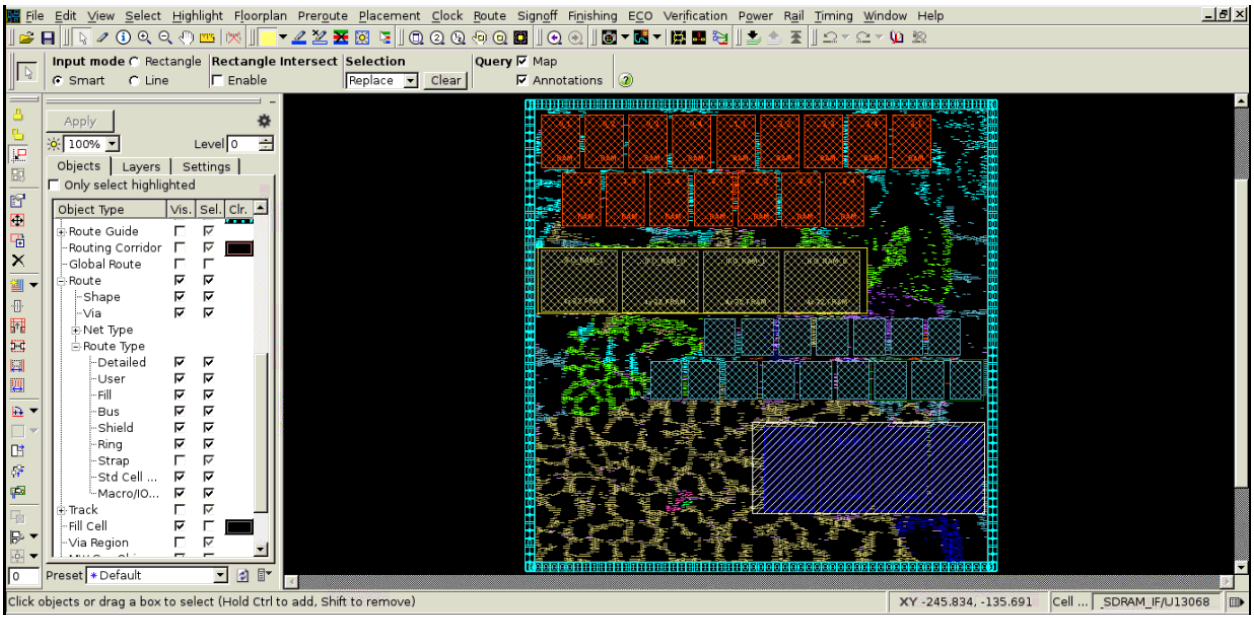

Figure:8 Standard cell view without PG nets

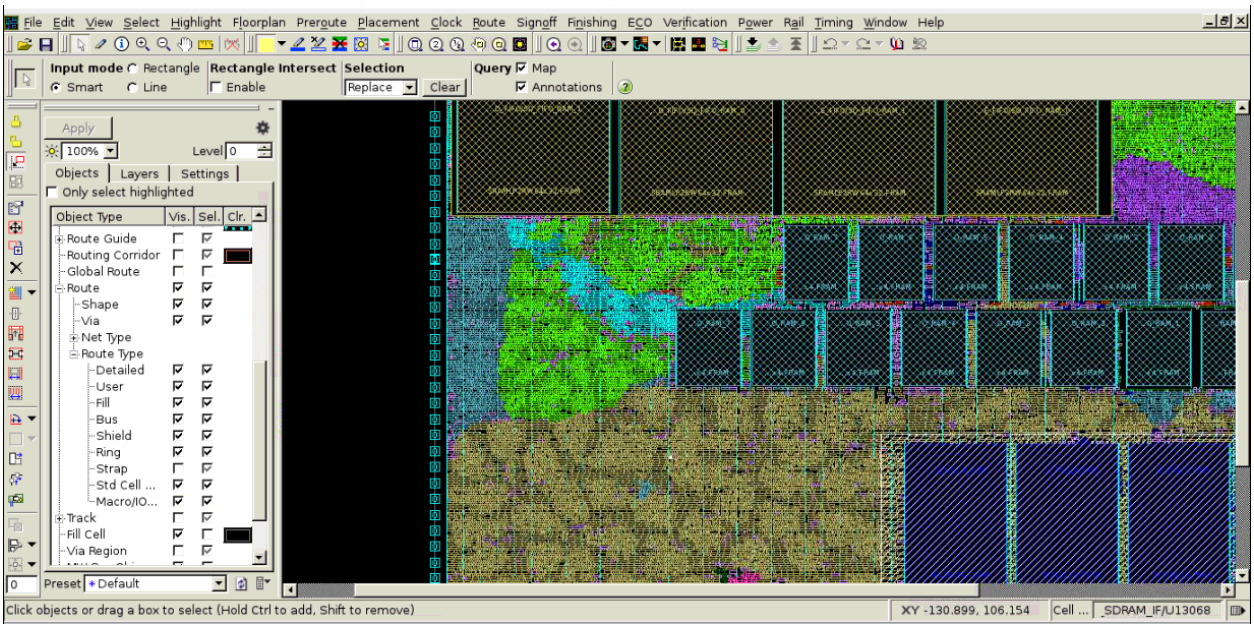

Figure:9 Standard cells distribution view

Clock Tree Synthesis is the concept of automatic insertion of buff/invs along the clock path to balance the clock skews. Most clock network synthesis algorithm use the summation or product configuration during the cock buffer insertion. Synopsys ICC uses **clock_opt** command to build the clock. This design has 3 master clock which can be reported using **report_clocks**. All three clock trees with their levels are shown in the following figures.

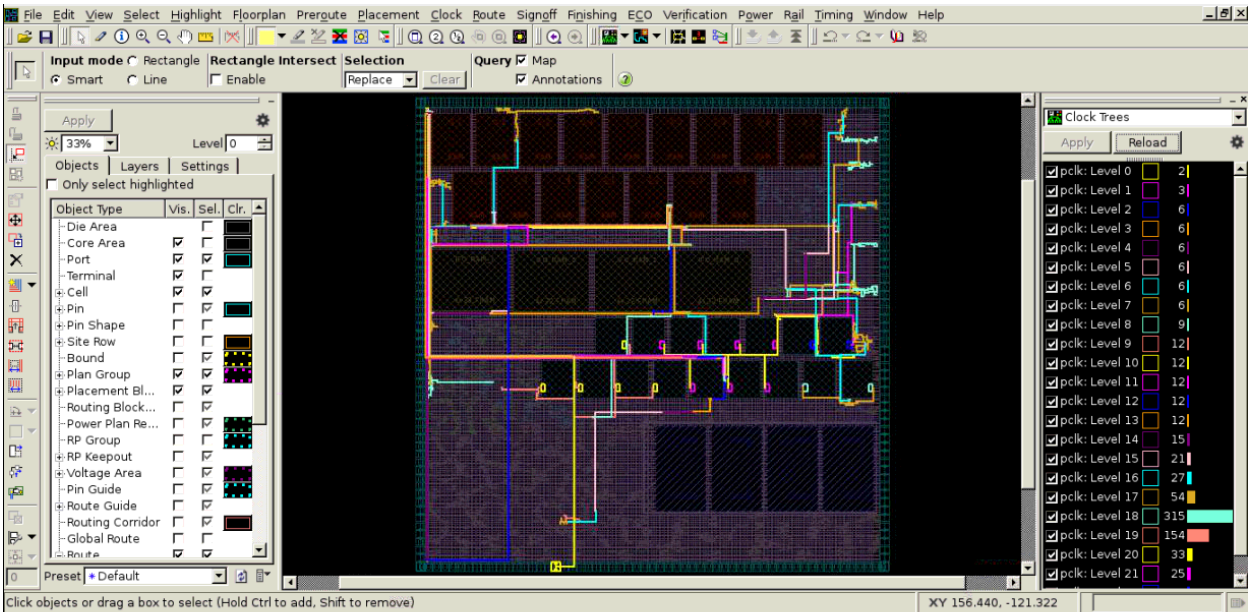

Figure:10 Master Clock 1 distribution

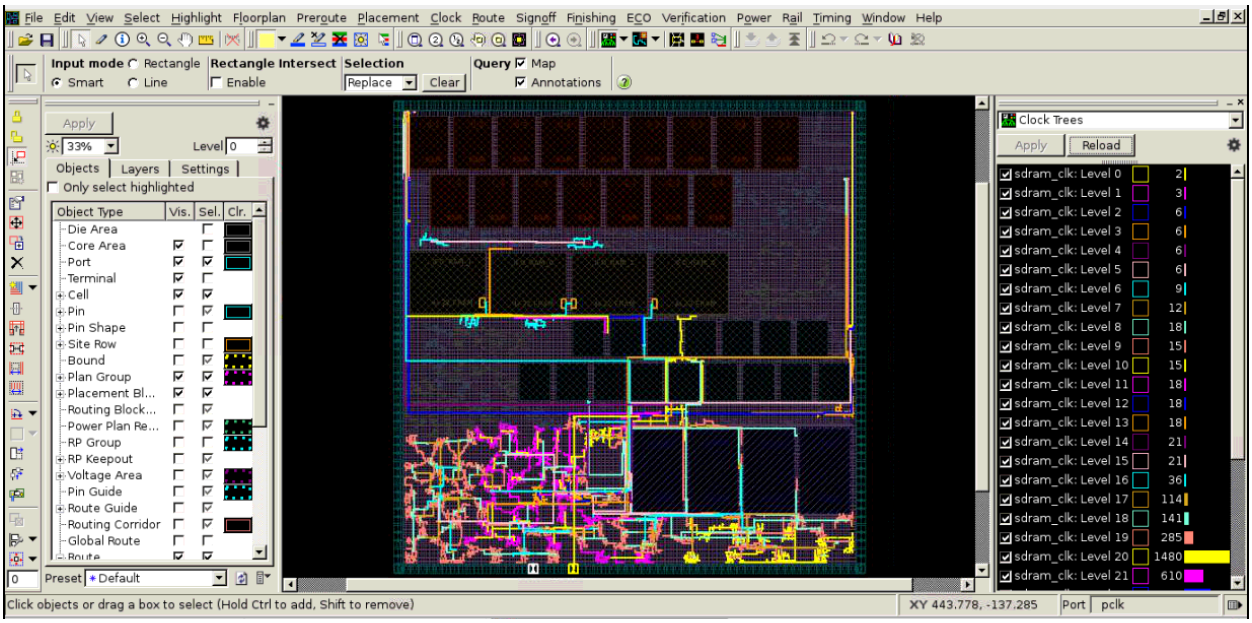

Figure:11 Master Clock 2 distribution

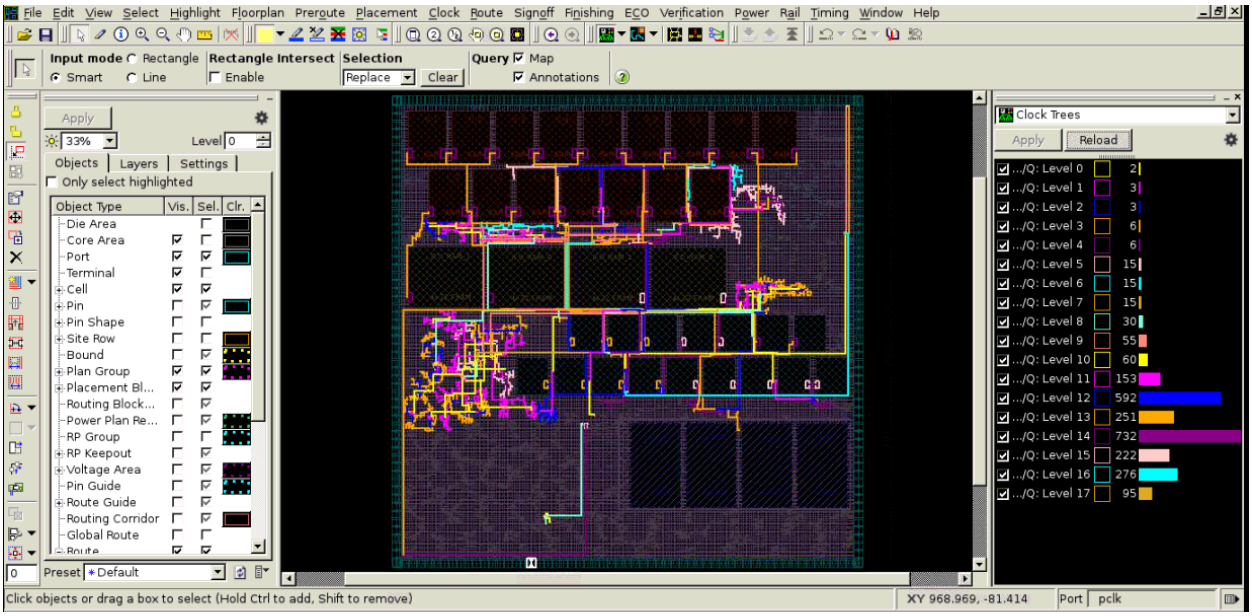

Figure:12 Master Clock 3 distribution

After completion of CTS and power analysis routing was performed to extract the parasitic data from the real routing of design. The purpose of SPEF generation is to do STA and simulation. Before starting routing, there are some sanity checks which need to be performed to make sure that the design is ready for routing. Commands like **check_mv_design**, **check_physical_design** (-pre_route_stage) and **check_routeability** are used to perform this checks. There are different routing algorithm but ICC uses the Z-route algorithm for the routing. The zrt route was done in four stages: global route, track assignment, detail route and search & repair. The next step was to generate various reports and verify the ZRT route & LVS. Finally, GDS, DEF and netlist were written for further phases.

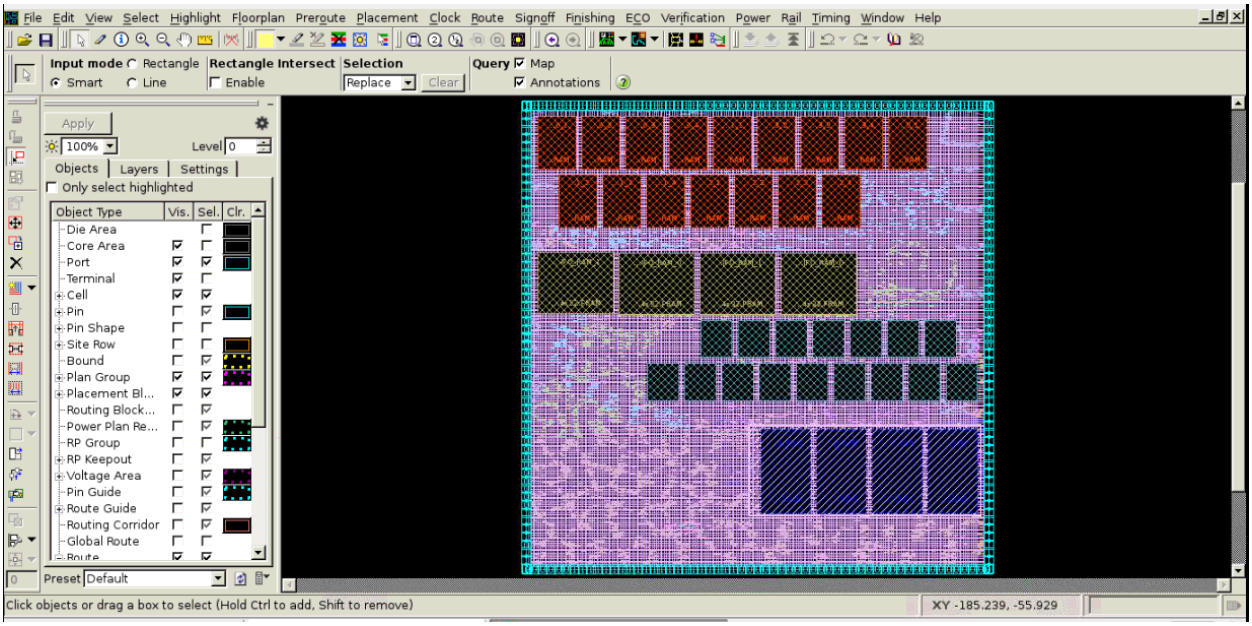

Figure:13 Routing completed

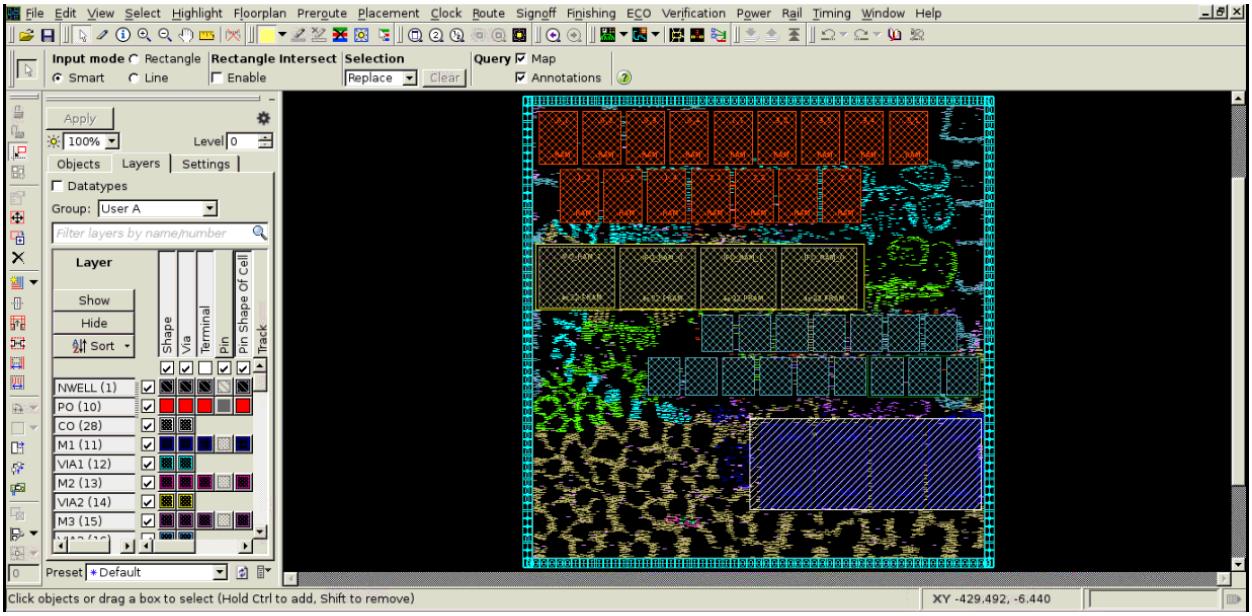

Figure:14 Routing without PG nets

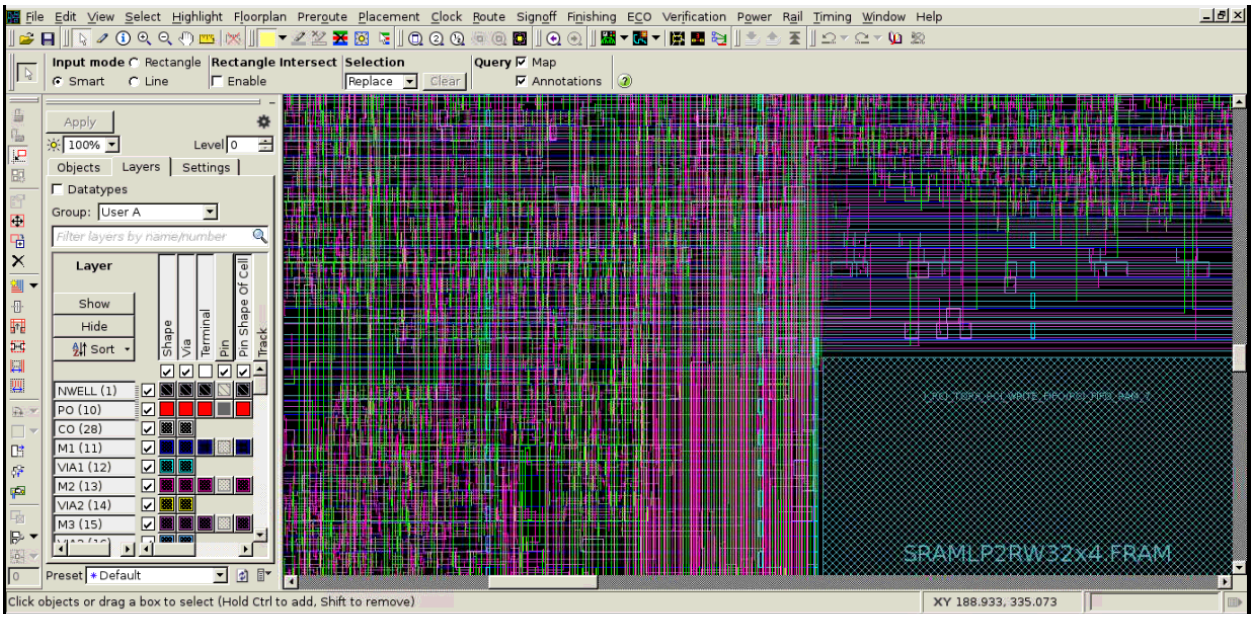

Figure:15 Routing layers

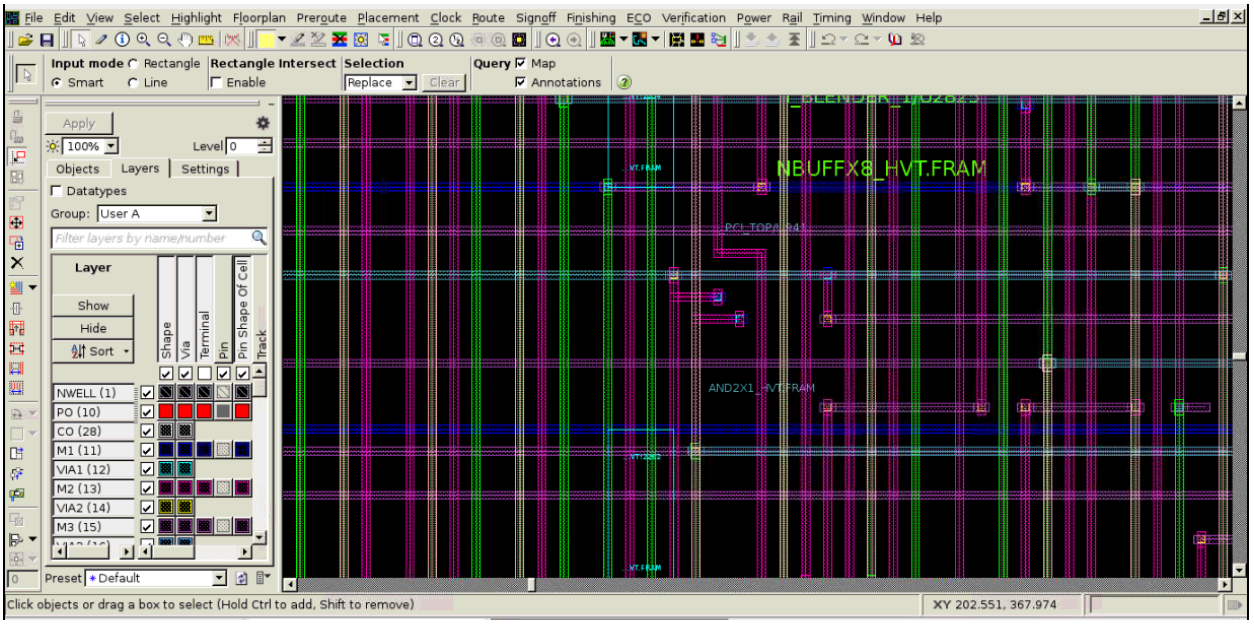

Figure:16 A closer view on metal layers

Phase: 4 **Parasitic Extraction** .nxtgrd | milkyway database | configuration file

Parasitic extraction is by analyzing each net in the design and assign the R & C values using pattern matching. The StarRC was used for SPEF generation. This tool uses .nxtgrd (New Extraction Generic Regression Database) and tech mapping files with design database. Control extraction function written in ASCII for was given as input to extractor to generate SPEF file. This step calculates all routed net capacitances and resistances for delay calculation done during timing analysis. The above-mentioned files were used on StarRC to generate the SPEF for for static timing analysis.

Phase: 5 **Static Timing Analysis** Physical netlist | SPEF | SDC

Static timing analysis was performed on the Primetime. After giving inputs to the tool and linking timing & physical library, the **read_paracitics** command was used to check if any missing annotation on the nets. This command did not shown any error which assured that there was no mismatching between .v and .spef. The option for including capacitive coupling was also true to take coupling capacitance into account while delay calculation.The internal and boundary nets were reported for annotation using **report_annotated_pararctics** command. As hold violations have top priority to fix, the best corner SPEF was generated using Cmin NXTGRG file for the inspection of worst path. The following figures show hold violation and the worst path associated with the violation generated by best corner spef file.

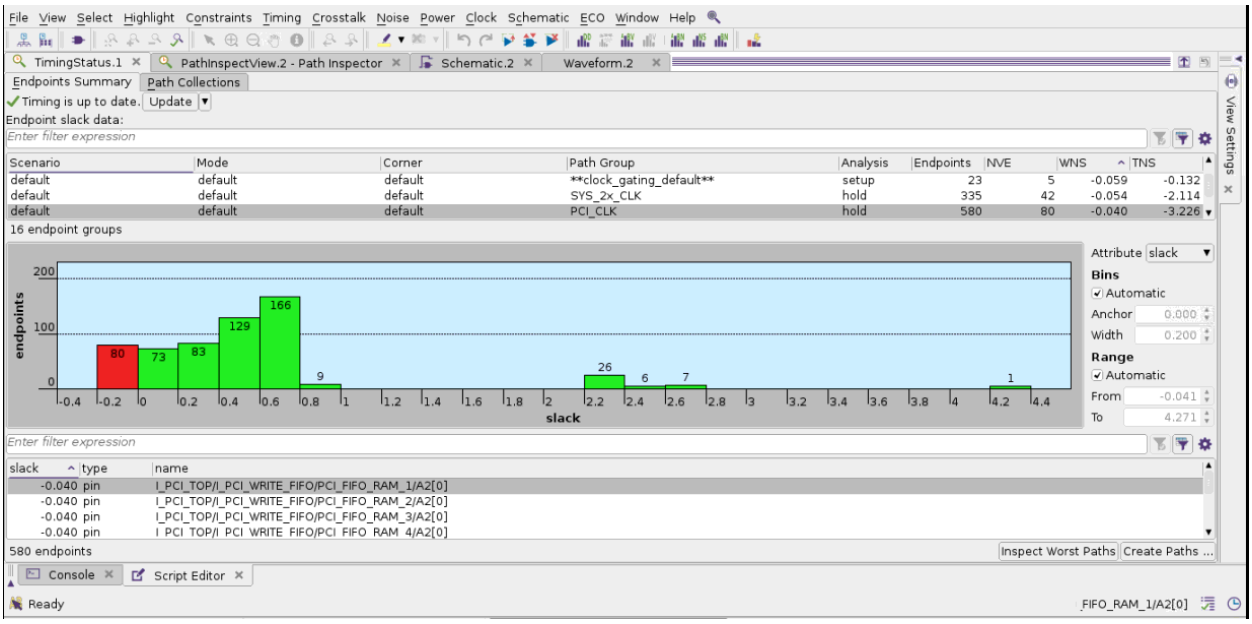

Figure:17 Setup/hold violations

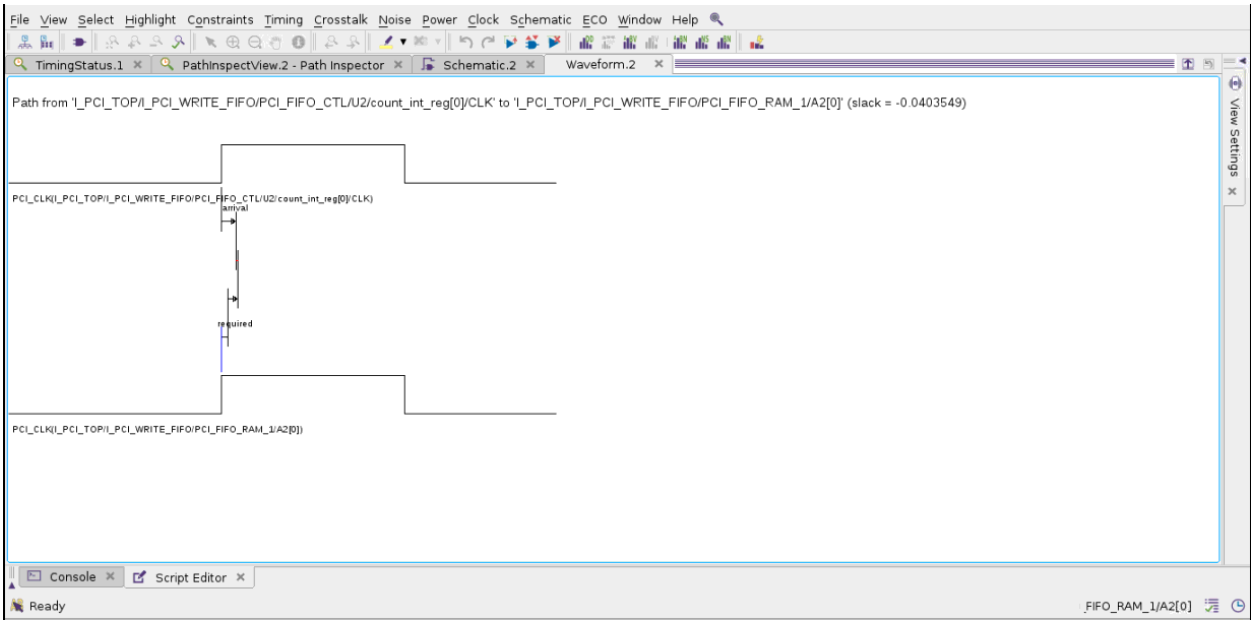

Figure:18 Hold violation waveform

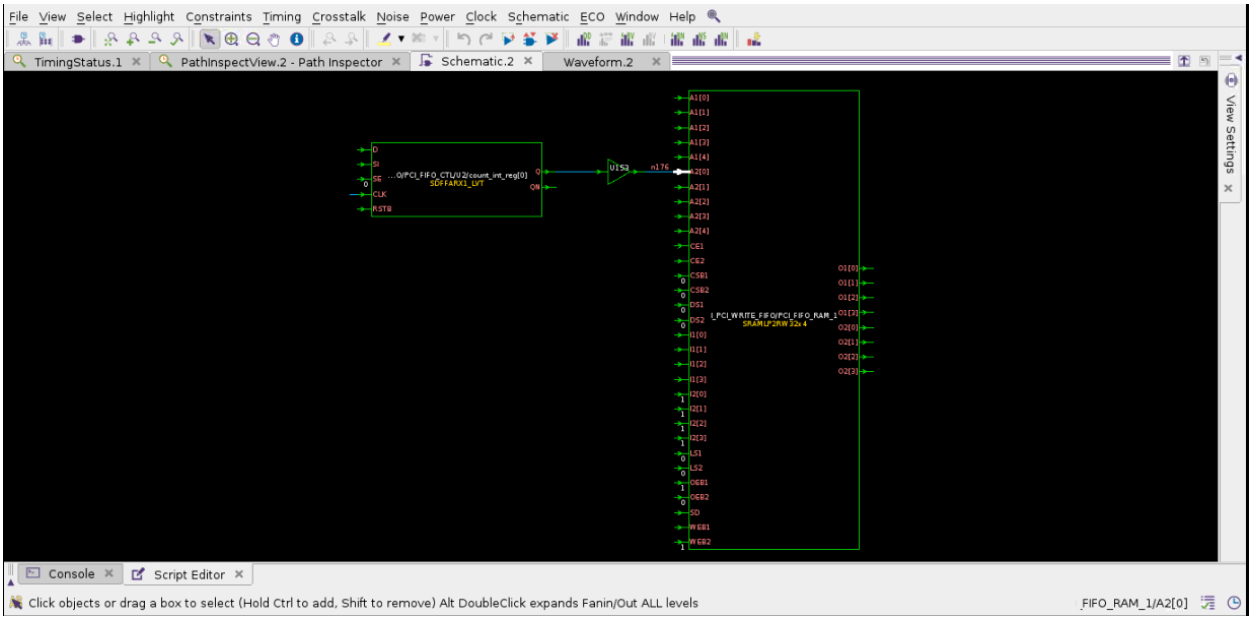

Figure:19 Hold violation path

Phase: 6 **Timing ECO Generation** Primetime_eco.tcl | SPEF | physical netlist

The purpose of this phase of the project is to implement the setup/hold and logical DRC ECOs on the routed netlist. Fixing ECO is an iterative procedure and it reduces the violations in the designs. ECOs were generated from pt_shell in form of TCL script and implemented on ICC. A single iteration can be define as follows-

ZRT Routing —> Physical Netlist (.v) —> SPEF(best corner) —>Primetime fix (logical DRC, hold) -—> ECO TCL generation —>Open Routed Database on ICC —> Implement ECOs on ICC —> Check Legality —> ZRT Route —>Save MW Cell —> Physical Netlist (.v)—> SPEF (worst corner)—>Primetime fix (logical DRC, setup) -—> ECO TCL generation —>Open Routed Database on ICC —> Implement ECOs on ICC —> Check Legality —> ZRT Route —>Save MW Cell—> Physical Netlist (.v)

…… **Continue for hold and setup again.**

This is the end of Project Description !