

Workshop Report: RTL to GDSII Flow

Trainers: VLSI Minds (Chethan & Team)

Dates: 21/04/26 to 24/04/26 and 27/04/26

The workshop provided intensive hands-on training on the digital backend flow, covering everything from functional verification of RTL code to the generation of the final GDSII stream-out file for fabrication. The primary design used for the flow was an SRAM Memory block (Depth=16, Width=8).

The workshop utilized the Cadence toolchain within a Linux/Shell environment:

1. Xcelium: Used for RTL Design and Functional Verification.
2. Genus: Used for Synthesis (converting RTL to Gate-Level Netlist).
3. Innovus: Used for Physical Design (Floorplanning, Placement, CTS, and Routing).
4. TCL/Shell Scripting: Used for tool invocation and design automation.

Day1: 21/04/26: RTL & Verification and Synthesis

The workshop began with the trainers introducing the SRAM Memory architecture. They emphasized the importance of a clean directory structure before even opening a tool. They taught how to set up environment variables and source the necessary Cadence licenses. The initial focus was on writing robust Verilog and using Xcelium to verify that our logic was "functionally correct" before proceeding to the hardware implementation stages.

The initial phase focused on directory setup and functional verification. Activities included:

- Creating structured directories: RTL_Design, RTL_Verification, Synthesis, and PNR.
- Verilog modeling of the SRAM memory signals (clk, rst, wr_en, wdata, rdata).
- Launching Xcelium ("xrun") for simulation and using waveform viewers to verify logic.

The trainers then transitioned into Logic Synthesis using Genus. A significant portion of the teaching was dedicated to Design Constraints (.sdc). We learned that synthesis isn't just about converting code to gates, but about optimizing for the "PPA" (Power, Performance, and Area) targets set by the technology library. Then Focus shifted to transforming the behavioral code into a structural netlist:

- Input preparation: RTL files (.v), technology libraries (.lib), and constraints (.sdc).
- Design elaboration and schematic generation.
- Quality of Result (QoR) report generation, specifically analyzing area, gate count, and timing.

Day 2 & 3: 22/04/26 - 23/04/26 : Physical Design -Floor Planning

I. Physical Design Initialization

- Environment Setup: The process began by creating a dedicated PNR (Place and Route) directory structure with subfolders: inputs, outputs, reports, scripts, and design.
- Tool Invocation: Innovus was invoked within the PNR directory to begin the physical implementation.

II. Essential Design Inputs

To start the PNR flow, the following design-specific files were required:

- Gate-Level Netlist (.v): The output from the synthesis stage (Genus).
- Design Constraints (.sdc): To define timing, clock definitions, and input/output delays.
- Scan Def: Required specifically for DFT (Design For Test) teams.
- UPF/LPF: Files for low-power design requirements.
- MCMM (Multi-Corner Multi-Mode): To ensure the design holds up across different operating conditions.

III. Technology & Library Files

The trainers emphasized the importance of the technology files provided by the foundry:

- Liberty Files (.lib): Contains timing, power, and functionality data for the standard cells.
- LEF (Library Exchange Format): * Technology LEF: Contains routing rules, metal layers, and via definitions.
 - Cell LEF: Provides the "abstract" view of the cells (physical boundaries and pin locations) used during placement and routing.
- DEF (Design Exchange Format): Provides the layout exchange format.
- Captable/Tlu+: Used for RC estimation and parasitic extraction to ensure accurate timing analysis.
- Direct Library Loading: In Cadence Innovus, .lib files can be loaded directly, whereas other tools (like those from Synopsys) may require conversion into a specific data model (like NDM).
- DEF Utility: The trainers explained that the .def file is crucial because it gives the tool a starting point for floorplanning, often including port placements and die boundaries.
- Standard Cell Varieties: The importance of using different threshold voltage cells (HVT, SVT, LVT) to balance the trade-off between leakage power and switching speed.

V. Reporting and Analysis (Quality of Result)

- Before moving to the next stage, we practiced generating QoR (Quality of Result) reports.

- Key reports included: report_qor, report_gates, report_area, report_power, and report_timing.
- The use of TCL scripting (e.g., get_db commands) was introduced to query the design database for specific information, such as the number of sequential vs. combinational instances.

FloorPlanning:

- Initializing the design with LEF and netlist files.
- Defining core/die area, aspect ratio, and utilization (set at 0.7).
- Power Planning: Creation of VDD/VSS rings, stripes, and rails to ensure power integrity.
- Port Placement: There are two methods to do port placement , side method and coordinate method . Ports are placed between core area and die area. We use side method , side 1 input ports and we go clockwise to place them.
- Physical Cells: There are 7 types of physical cells:
 - End cap cells(WPE)
 - Tap cell(latch up problem)
 - Tie cell(logic 1 and logic 0 is required)
 - Decap cell(boosting voltage to cell)
 - Filler cell
 - Spare cell(due to engineering change)
 - Antenna cell(metal jumper)

Day 4: 24/04/26 : Placement

Placement of standard cells is the siterows in core area. Placement reduces congestio problem. Tracks are used to route metals.

Placement can be based on:

- Timing driven placement
- Area driven placement
- Power driven placement
- Congestion driven placement

Types of placement:

- Coarse placement(tentatively place std. cells) (Global placement) (Rough placement)
- Legalize placement(legalize onsite rows)
- Initial placement
- Final placement

Critical backend stages to ensure clock reliability and connectivity:

- Clock Tree Synthesis (CTS): Balancing skew and latency using algorithms like L-Tree, H-Tree and Fishbone algorithm.
- Skew should be zero or positive. To balance skew, we add clock buffers and clock inverters on the clock path.

Day 4: 27/04/26 : Routing & GDSII

The final phase focused on connectivity and manufacturability:

- Routing: Executing global and detail routing using “routeDesign”; and verifying DRC/LVS compliance.
- Timing Sign-off: Using STA to fix hold/setup violations post-routing.
- Tape-out: Final generation of the GDSII file (Graphical Data Stream) as the blueprint for fabrication.

MACRO floorplanning, placement and routing:

Macro placement guidelines:

1. Don't overlap the macros.
2. Place macro at core boundary.
3. Don't place macro that should block the ports.
4. Don't place the macros with notches.
5. Place the macros according to hierarchy.
6. Don't place the macros with criss cross connections.

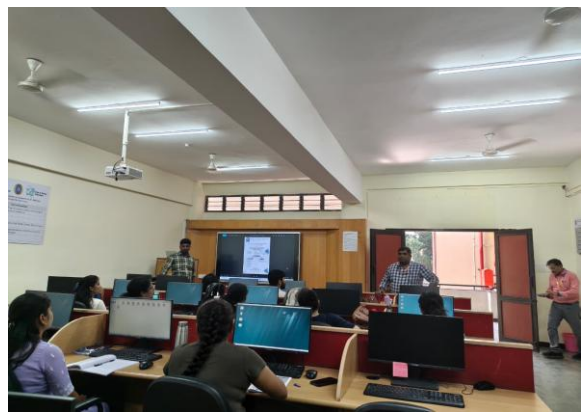
Then, the same process was followed for floorplanning, placement and routing of the macros.

Technical Takeaways

- Standard Cell Flavors: Strategic use of HVT, SVT, and LVT cells to optimize power and speed.

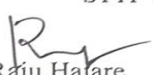
- Physical Cells: Implementing End-caps, Well-taps, and Decap cells for reliability.
- Metal Layers: Understanding Horizontal (Odd layers) vs. Vertical (Even layers) routing rules.
- Automation: Heavy emphasis on TCL scripting to automate the PNR flow and generate consistent reports.

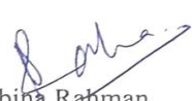
Under the guidance of the Chethan team, this workshop provided a comprehensive understanding of the VLSI design cycle. The hands-on exposure to Cadence tools and industry-standard scripting prepares us for professional roles in Silicon Engineering.



STTP Coordinators


Dr. Banuprakash R
PG-VSD Coordinator
Assoc. Professor, ECE


Dr. Raju Hajare
Professor, ECE


Dr. Sabina Rahman
Assoc. Professor, ECE


Dr. Jagannath K B
Assoc. Professor, ECE


HoD ECE
Dr. A Shobha Rani