Project on

Physical Design Implementation of Serial Peripheral Interface Using Q flow

Name: Yemireddy Vignesh Reddy

Course: VLSI PD Internship

Batch: B2B PDI 03

Table of contents

Introduction

Q flow is an open-source VLSI design flow tool that automates RTL-to-GDSII flow for digital IC design using open-source EDA tools. It helps you go from Verilog HDL to layout (GDSII) for fabrication, making it a valuable tool for academic, learning, and low-cost ASIC prototyping. Supports RTL synthesis, logic optimization, placement, routing, DRC, LVS, and GDSII generation. Uses Yosys, Graywolf, Qrouter, Magic, Netgen internally. Compatible with FreePDK45, Sky130, etc. Command-line driven, suitable for automation and scripting. Lightweight and fast for small to medium digital blocks. A digital synthesis flow is a set of tools and methods used to turn a circuit design written in a high-level behavioural language like Verilog or VHDL into a physical circuit, which can either be configuration code for an FPGA target like a Xilinx or Altera chip, or a layout in a specific fabrication process technology, that would become part of a fabricated circuit chip. Several digital synthesis flows targeting FPGAs are available, usually from the FPGA manufacturers, and while they are typically not open source, they are generally distributed for free (presumably on the sensible assumption that more people will be buying more FPGA hardware).

Tools in Q flow:

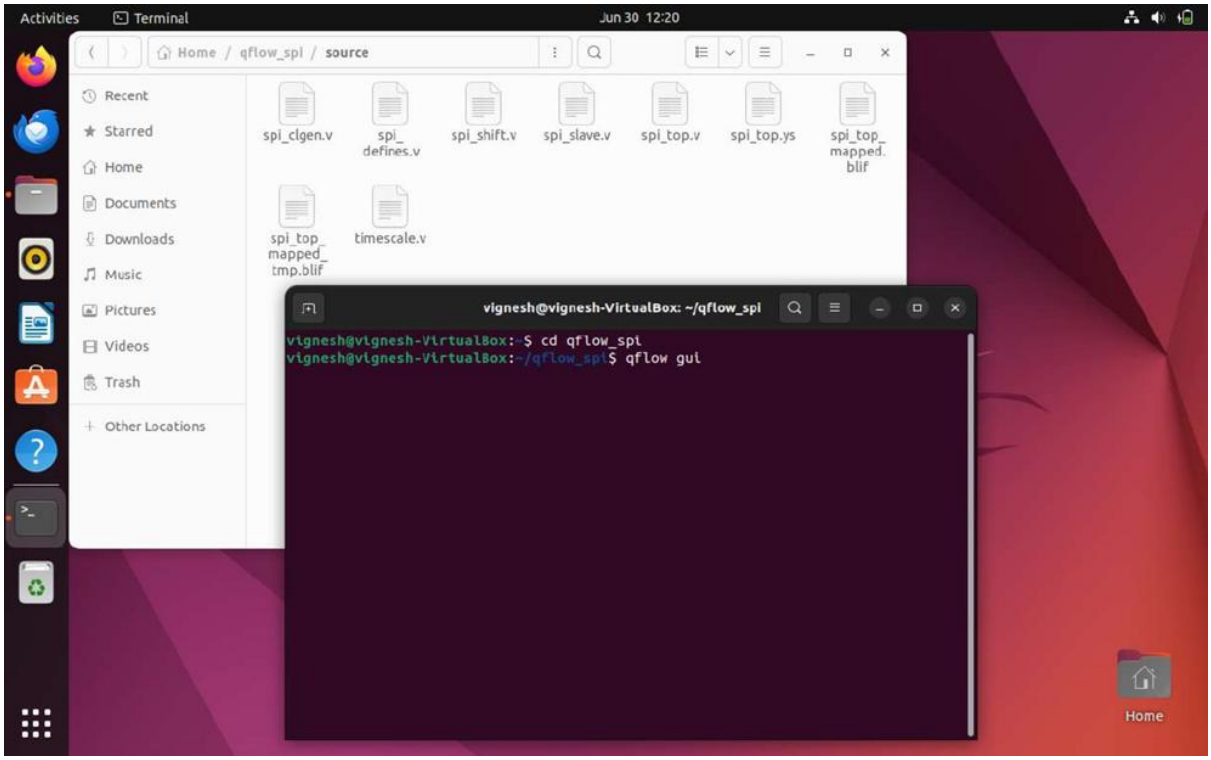
Stage	**Tool**	**Purpose**
RTL Synthesis	**Yosys**	Converts Verilog HDL to gate-level netlist
Logic Optimization	**ABC**	Optimizes logic for area/speed
Placement	**Graywolf**	Places standard cells on the chip area
Routing	**Qrouter**	Connects the placed cells with metal layers
DRC / LVS	**Magic, Netgen**	Design rule checks, layout-vs-schematic verification
GDSII Generation	**Magic**	Exports GDSII for fabrication

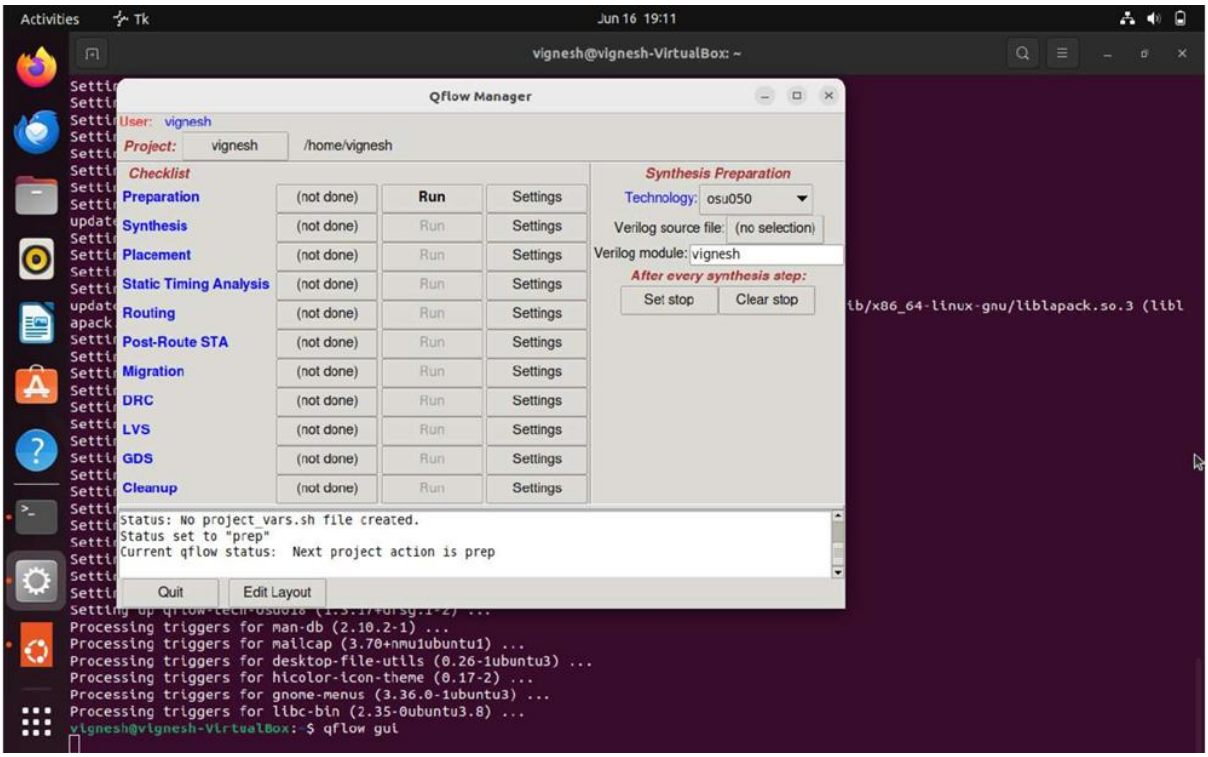
Q flow is a complete tool chain for synthesizing digital circuits starting from Verilog source and ending in physical layout for a specific target fabrication process. In the world of commercial electronics, digital synthesis with a target application of a chip design is usually bundled into large EDA software systems like Cadence or Synopsys. As commercial electronics designers need to maintain cutting-edge performance, these commercial toolchains get more and more expensive and have largely priced themselves out of all but the established integrated circuit manufacturers. Q flow is a hands-on, practical tool to learn and practice VLSI backend design using open-source tools and PDKs. It will bridge your RTL design understanding to physical implementation, making you comfortable with the VLSI flow used in industry.

Invoke the tool

To invoke the tool, open the terminal in the Oracle Virtual Box.

For setting up the Qflow Gui:





Q flow GUI:

Qflow GUI is the graphical frontend for Qflow ASIC synthesis flow, allowing RTL-to-GDSII flow without typing each terminal command manually.

For this project first we need to create the directory by naming it as **"qflow_spi"** in which we need to have the Verilog RTL files in it. To create the directory, open the terminal give the command **"mkdir"** along with the name of the file like **"mkdir qflow_spi".** Then give the command **cd qflow_spi** to open the directory and then **qflow gui.**

First, we need to select the Technology as **osu018.** After selecting this we need to select the Verilog Source File as **"qflow_spi".** Then we need select the Verilog module as **"spi_top.v".**

By clicking the run option, we can invoke the individual tool and if the tool executes without showing any error, then we can see as OKAY, if there is any error then it shows FAIL.

Preparation

The **Preparation step** is the **first stage in Qflow's RTL-to-GDSII digital flow**. It prepares the design directory and necessary files. It reads your RTL (Verilog) code. It synthesizes a preliminary netlist using Yosys. It generates initial files required for further steps like synthesis, placement, and routing.

In **Qflow GUI**, you can execute it by:

Selecting your project then by clicking **Preparation** and click **Run**.

Purpose of the Preparation Step

- To **check your Verilog files for syntax errors**.
- To **convert RTL (register-transfer level) to a preliminary gate-level netlist**.
- To **ensure library and constraint files are correctly connected**.
- To **set up the environment with necessary technology parameters** before synthesis.

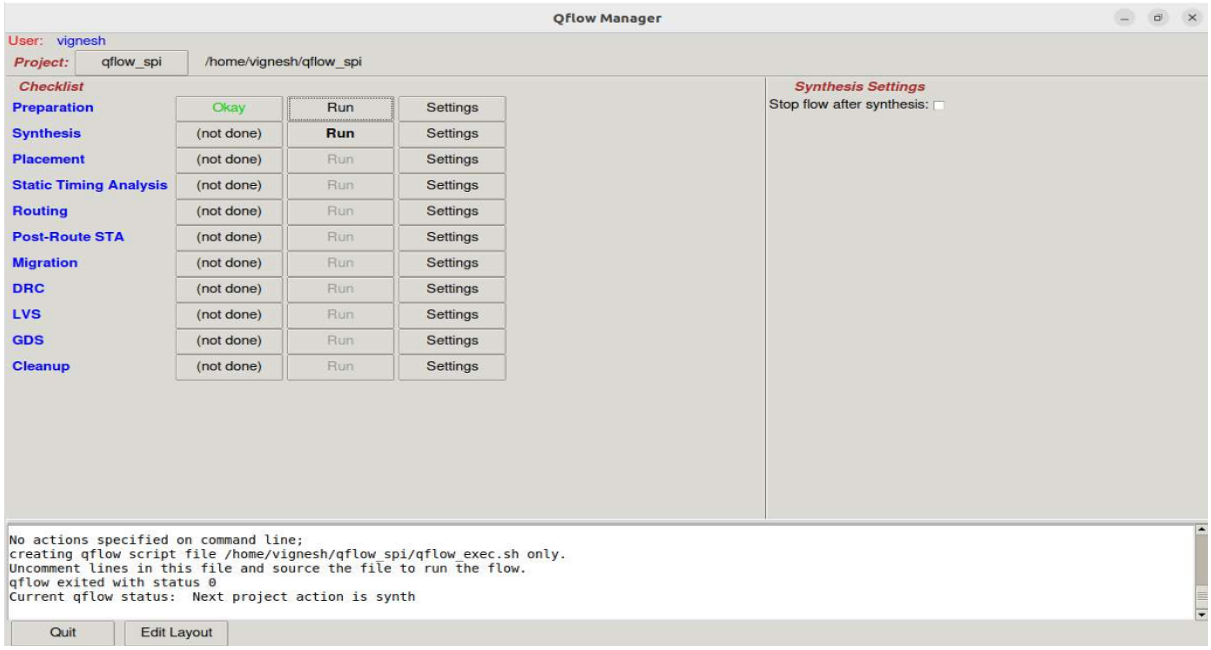
The **Preparation step in Qflow GUI** is the **first and foundational stage** of the Qflow RTL-to-GDSII digital ASIC design flow. This step is responsible for setting up your design environment, checking your Verilog RTL for syntax correctness, and preparing your project workspace for the subsequent synthesis, placement, and routing steps. You initiate this step in the Qflow GUI by selecting your project and clicking **Preparation**, which automatically organizes your design files, sets up necessary directories, and performs initial parsing of your design.

Internally, during the preparation step, **Qflow uses the Yosys synthesis tool in its parsing mode** to read and elaborate your Verilog RTL files. This helps identify syntax errors, missing modules, unconnected signals, or potential inferred latches early, allowing you to correct your design before moving forward. The preparation step ensures that the top module is correctly recognized, the design hierarchy is consistent, and any basic issues that may hinder synthesis are caught at this stage, reducing debugging time later in your design flow.

Additionally, this step configures the environment by setting up **necessary project variables and environment paths**, including linking your project with the required **standard cell library files** (`.lib` for timing and logical information, . lef for physical dimensions, and. spi for simulation if needed). The preparation step checks the accessibility and correctness of these library files according to your specified technology node (e.g., `osu018`, `osu035`), ensuring that Qflow will be able to map your design to the cells during synthesis and placement.

Another key aspect of the preparation step is the generation of the **initial `.blif` file**, which is a Berkeley Logic Interchange Format representation of your RTL design. This `.blif` file serves as the input for the subsequent synthesis step, where it will be optimized and mapped to technology-specific cells. The preparation step also generates logs detailing the parsing and elaboration process, providing insights into any warnings or errors encountered during RTL parsing.

In summary, the **Preparation step in Qflow GUI organizes your project environment, verifies your Verilog RTL syntax and hierarchy using Yosys, checks library and technology file paths, and generates an initial `.blif` representation of your design** to be used in synthesis. This step ensures your design environment is clean, consistent, and ready for synthesis, providing a smooth transition into the physical design phases of your VLSI ASIC flow.



Synthesis

The Synthesis step in Qflow GUI is a critical stage that transforms your high-level Verilog RTL design into a gate-level netlist using your target technology's standard cells. After the *Preparation* step, where your design environment and initial files are set up, the synthesis stage performs logical synthesis and technology mapping, ensuring your design can be physically realized in silicon. This step is executed within the Qflow GUI by selecting your project and clicking Synthesis, which triggers the underlying tools to process your design files automatically.

The **Synthesis step** in Qflow:

1. Converts **RTL (Verilog)** to an **optimized gate-level netlist** using your standard cell library.
2. Maps your design to **physical cells** (NAND, NOR, Flip-Flops, etc.).
3. Optimizes for **area, timing, and logic reduction**.
4. Prepares the netlist for **floor planning, placement, and routing**.

In **Qflow GUI**, you run it by:

Selecting your project then by Clicking **Synthesis**, Click **Run**.

Purpose of the Synthesis Step

- Translate your **high-level Verilog** into **hardware using standard cells**.
- Minimize **logic depth and area**.
- Apply **technology constraints** like cell delays and drive strengths.
- Prepare a **clean netlist** for physical design steps.

Yosys Tool:

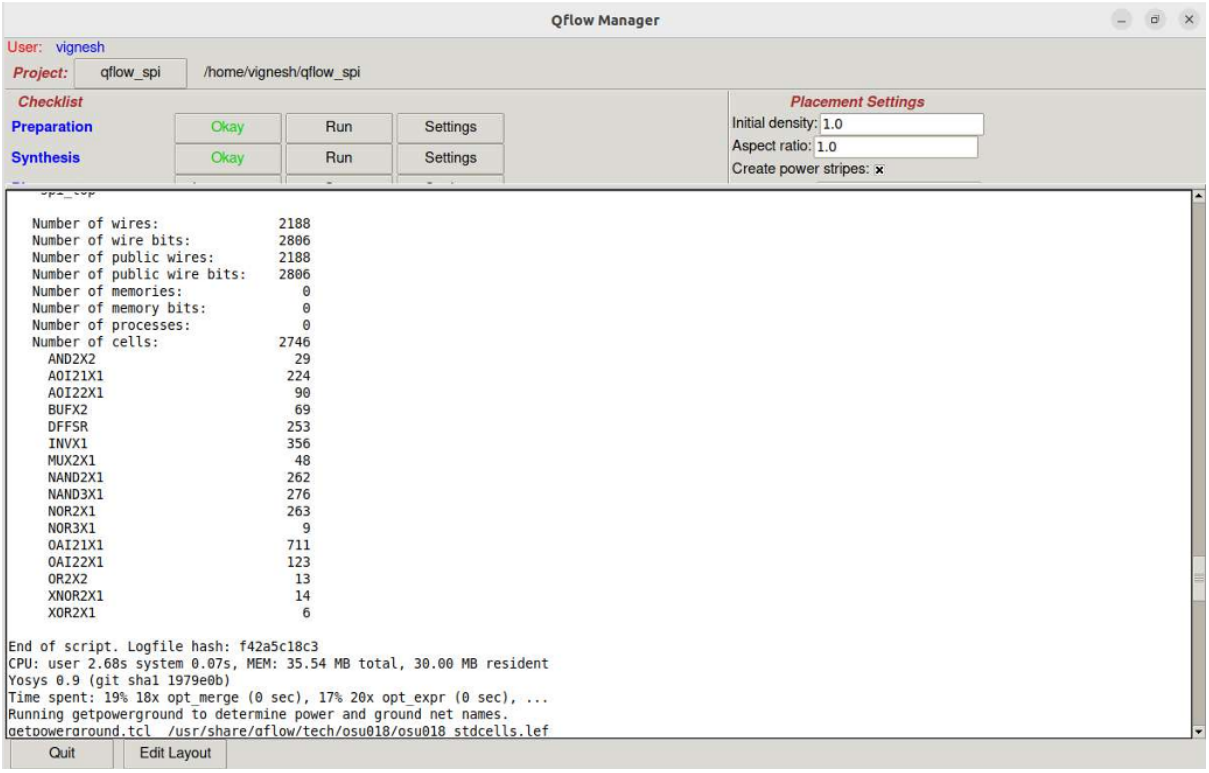
Yosys is an open-source framework for RTL (Register Transfer Level) synthesis, meaning it translates hardware descriptions written in languages like Verilog into a gate-level netlist that can be implemented on physical hardware. It's designed to be flexible and extensible, making it suitable for various synthesis tasks, including mapping to standard cell libraries for ASICs and different FPGA architectures like Xilinx 7-Series.

Internally, the synthesis step uses **Yosys**, an open-source synthesis tool integrated within Qflow, to read your Verilog design, parse it for correctness, and translate it into a representation of logic gates that implements the same functionality described in your RTL. During this process, Yosys first elaborates your design, checking for syntax errors, unconnected signals, and design hierarchy consistency. It then applies **logic optimization** to reduce redundant logic, minimize area, and simplify combinational paths while preserving functionality. This includes constant propagation, dead code elimination, and logic simplification to optimize your design before technology mapping.

Once optimization is complete, Yosys performs **technology mapping** using the abc tool, which is called internally. This stage maps the generic gates from your synthesized logic to **specific cells available in your standard cell library** (for example, osu018.lib or osu035.lib), ensuring that your synthesized design can be physically implemented using the library's available NAND, NOR, flip-flops, and other gates. This mapping also considers timing and area optimization based on your constraints and the technology library, allowing you to analyze the critical paths in your design during the subsequent physical design stages.

After the synthesis step completes, Qflow generates several important output files. The primary output is the **gate-level .blif file**, which contains your design in a format readable by the placement and routing tools in the Qflow pipeline. Additionally, a **gate-level Verilog file (.v)** is created, which you can use for post-synthesis simulation and verification to confirm that your design's functionality remains correct after synthesis. A detailed **log file (. synth.log)** is also generated, which includes information about the synthesis process, any warnings or errors encountered, and reports on area and gate usage, helping you debug and refine your design.

In summary, the **Synthesis step in Qflow GUI converts your functional Verilog description into a technology-mapped, gate-level netlist ready for physical implementation**, using Yosys for logical synthesis and abc for technology mapping. It ensures your design is optimized for the targeted standard cell library while maintaining the intended functionality, setting the stage for placement, routing, and final layout generation in your VLSI design flow.



Placement

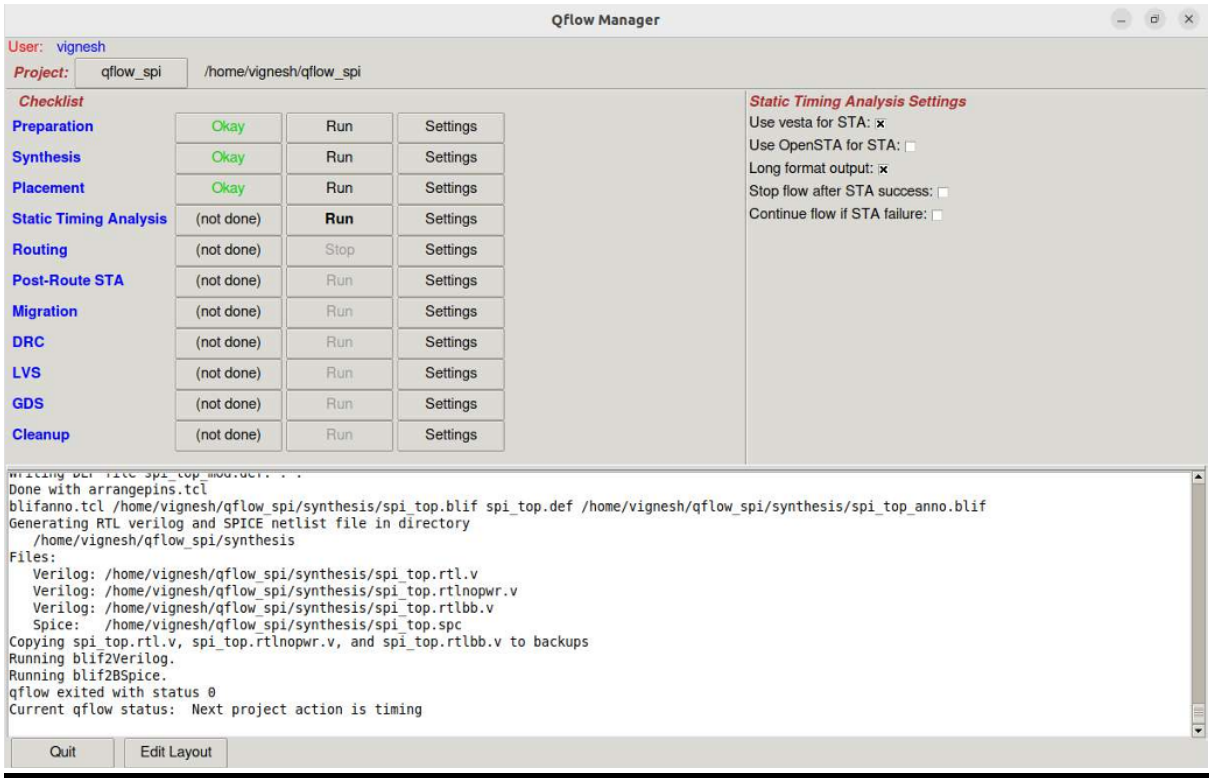
The **Placement step in Qflow GUI** is the stage where the synthesized, technology-mapped netlist from the previous step is physically arranged on the chip area before routing begins. After the synthesis step generates a gate-level netlist using standard cells, placement determines the **physical coordinates of each cell within the defined core area** of your ASIC layout, ensuring that the cells are positioned efficiently to minimize area, wire length, and potential timing issues during routing. You initiate this step in the Qflow GUI by selecting your project and clicking **Placement**, allowing the underlying tools to handle the complex task of cell arrangement automatically.

Internally, Qflow uses **Graywolf**, an open-source placement tool, to perform the placement task. Graywolf reads the. blif netlist generated by synthesis, along with the standard cell information and the technology's .lef file defining cell geometries and blockages, to calculate an optimal placement of each cell on the die. The tool uses a combination of **min-cut placement and iterative improvement algorithms** to reduce the total estimated wire length between connected cells while considering the area utilization and the physical design rules of the technology library. By minimizing wire length, the tool indirectly improves the overall timing performance and reduces potential congestion in later routing stages.

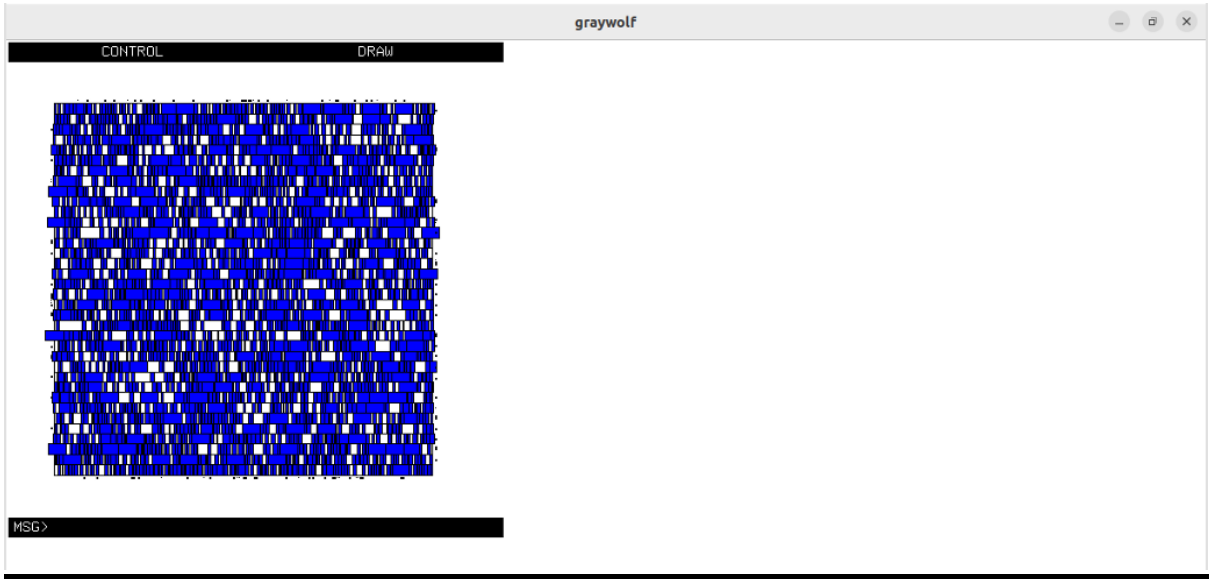
During placement, input constraints such as **IO pin locations, floorplan dimensions, and aspect ratios** (defined during your initial project setup or preparation) guide how the cells are spread across the die area. If any macro cells (larger predefined blocks like SRAM or IO blocks) are present, Graywolf positions them appropriately within the layout while placing standard cells around them. This step also checks for overlapping cells, ensuring each cell is uniquely positioned within the available space without violating the design's physical constraints.

After the placement process completes, Qflow generates several output files critical for your VLSI design flow. The key outputs include a **.cel file**, which contains the cell placement coordinates for each standard cell in the design, and a **.rc file** with placement results used by the subsequent routing step. Additionally, Qflow generates logs detailing placement statistics, such as estimated wire lengths, utilization, and any warnings encountered during placement, which you should review to confirm that your design is progressing correctly before routing.

In summary, the **Placement step in Qflow GUI physically organizes your synthesized netlist on the chip area using Graywolf to optimize cell positions while minimizing estimated wire length and ensuring design rule compliance**. This step is essential for achieving an efficient and routable layout, setting the stage for the routing step, where the actual physical connections between cells will be established, moving you closer to generating a complete GDSII layout for fabrication in your ASIC design flow.


Qflow Manager
User: vignesh
Project: qflow_spi
/home/vignesh/qflow_spi
Checklist
Preparation Okay Run Settings
Synthesis Okay Run Settings
Placement Okay Run Settings
Static Timing Analysis (not done) Run Settings
Routing (not done) Stop Settings
Post-Route STA (not done) Run Settings
Migration (not done) Run Settings
DRC (not done) Run Settings
LVS (not done) Run Settings
GDS (not done) Run Settings
Cleanup (not done) Run Settings
Static Timing Analysis Settings
Use vesta for STA:
Use OpenSTA for STA:
Long format output:
Stop flow after STA success:
Continue flow if STA failure:
Done with arrangepins.tcl
blifanno.tcl /home/vignesh/qflow_spi/synthesis/spi_top.blif spi_top.def /home/vignesh/qflow_spi/synthesis/spi_top_anno.blif
Generating RTL verilog and SPICE netlist file in directory
/home/vignesh/qflow_spi/synthesis
Files:
Verilog: /home/vignesh/qflow_spi/synthesis/spi_top.rtl.v
Verilog: /home/vignesh/qflow_spi/synthesis/spi_top.rtlnopwr.v
Verilog: /home/vignesh/qflow_spi/synthesis/spi_top.rtlbb.v
Spice: /home/vignesh/qflow_spi/synthesis/spi_top.spc
Copying spi_top.rtl.v, spi_top.rtlnopwr.v, and spi_top.rtlbb.v to backups
Running blif2Verilog.
Running blif2BSpice.
qflow exited with status 0
Current qflow status: Next project action is timing
Quit
Edit Layout

Graywolf:


graywolf
CONTROL
DRAW
MSG>

Static Timing Analysis

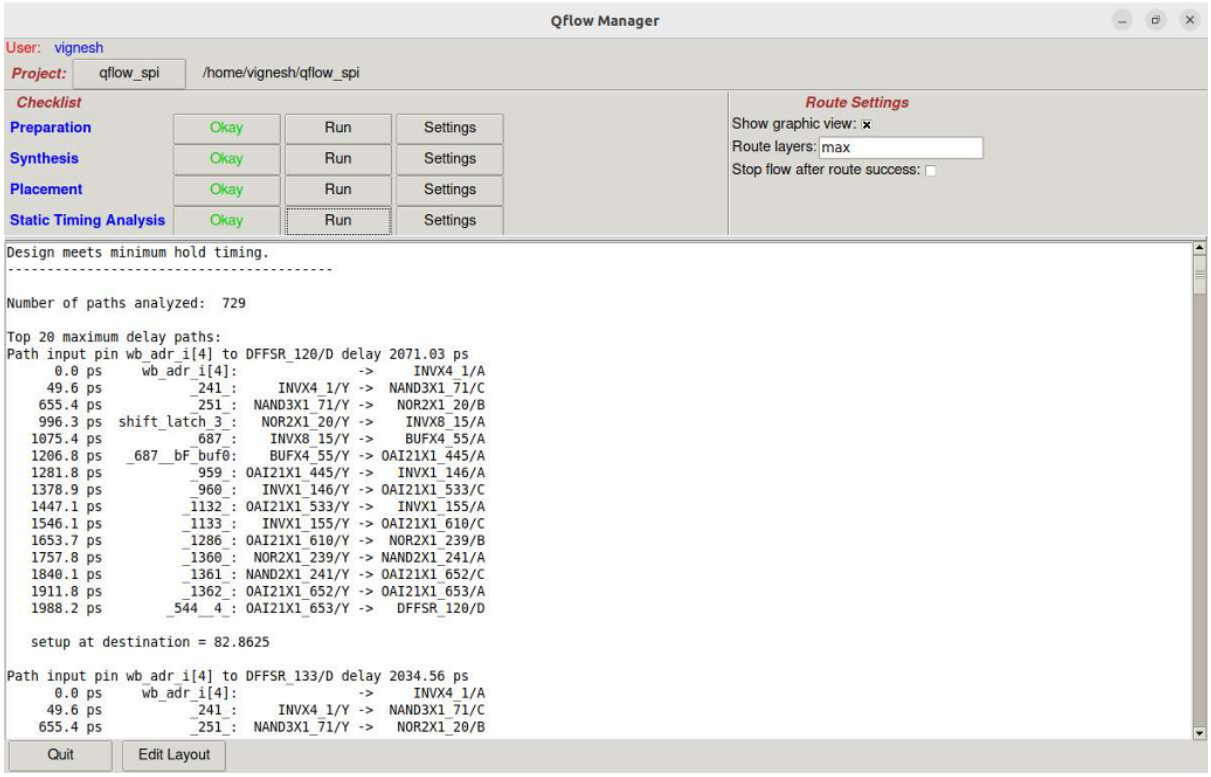
The **Static Timing Analysis (STA) step in Qflow GUI** is the stage where your design's **timing behavior is evaluated after placement and routing to ensure it meets the required performance and clock constraints**. Unlike dynamic simulation, STA does not require input vectors but instead systematically checks all possible timing paths in your design to identify critical paths and potential timing violations. You run this step in the Qflow GUI by selecting your project and clicking **Analysis** after routing, enabling automated timing verification using the integrated STA tools within Qflow.

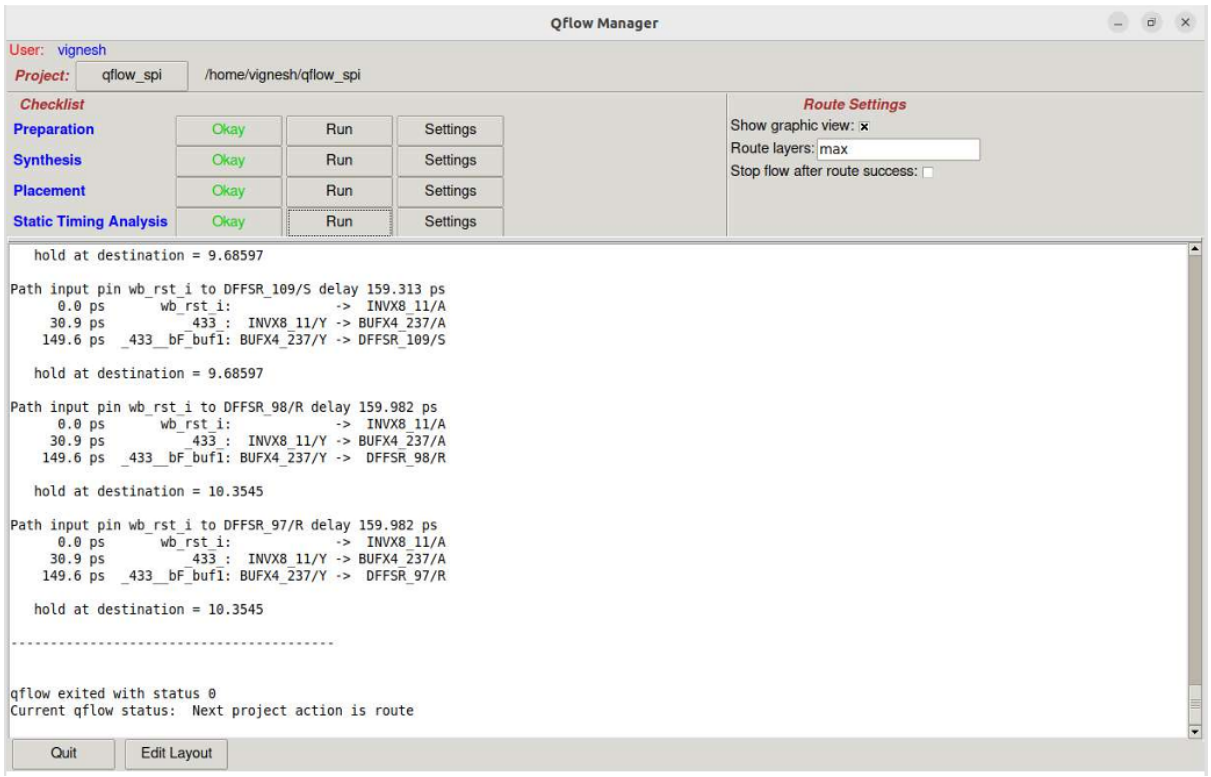
Internally, Qflow performs STA using **vesta**, an open-source timing analysis tool designed to work with the Qflow pipeline. Vesta analyzes your **final routed netlist**, using the .blif or .v gate-level netlist and the parasitic information extracted during the routing stage. It reads your **standard cell library's timing data (.lib)**, which contains cell delays and setup/hold timing characteristics, and the user-defined clock constraints specified during the preparation or synthesis stages. Vesta traverses all timing paths between registers, from input ports to registers, and from registers to output ports, computing the arrival times and required times to ensure your design meets setup and hold timing requirements.

During STA, the tool identifies **critical paths**, which are the longest paths between two timing endpoints in the design, as these determine the maximum operating frequency your design can achieve without timing failures. If the arrival time on any path exceeds the required time (determined by the clock period), a timing violation is reported. Similarly, hold time violations, where signals arrive too early, are also checked to prevent potential data corruption. These timing checks ensure that your design will function reliably at the intended clock frequency in silicon.

Upon completion, the STA step generates detailed **timing reports**, including a list of critical paths, maximum and minimum delays, slack values, and a clear indication of whether your design meets timing constraints. These reports are essential for identifying timing bottlenecks, allowing you to refine your design by optimizing critical paths, adjusting placement, modifying constraints, or even modifying your RTL if necessary to meet performance goals. Qflow's STA step provides a clear view of your design's timing health before moving toward generating the GDSII layout for tape-out or further physical verification.

In summary, the **Static Timing Analysis step in Qflow GUI systematically checks the timing of your fully placed and routed ASIC design using the vesta tool, leveraging the standard cell timing library and clock constraints to ensure your design meets its required frequency and has no setup or hold timing violations**. This step is crucial for verifying the operational correctness of your design at the intended clock speed and guarantees that your ASIC design is ready for final layout generation, functional simulation, and sign-off without risking timing failures in silicon.


Qflow Manager
User: vignesh
Project: qflow_spi
/home/vignesh/qflow_spi
Checklist
Preparation Okay Run Settings
Synthesis Okay Run Settings
Placement Okay Run Settings
Static Timing Analysis Okay Run Settings
Route Settings
Show graphic view:
Route layers: max
Stop flow after route success:
Design meets minimum hold timing.
Number of paths analyzed: 729
Top 20 maximum delay paths:
Path input pin wb_adr_i[4] to DFFSR_120/D delay 2071.03 ps
0.0 ps wb_adr_i[4]: -> INVX4_1/A
49.6 ps _241_: INVX4_1/Y -> NAND3X1_71/C
655.4 ps _251_: NAND3X1_71/Y -> NOR2X1_20/B
996.3 ps shift_latch_3_: NOR2X1_20/Y -> INVX8_15/A
1075.4 ps _687_: INVX8_15/Y -> BUFX4_55/A
1206.8 ps _687__bF_buf0: BUFX4_55/Y -> OAI21X1_445/A
1281.8 ps _959_: OAI21X1_445/Y -> INVX1_146/A
1378.9 ps _960_: INVX1_146/Y -> OAI21X1_533/C
1447.1 ps _1132_: OAI21X1_533/Y -> INVX1_155/A
1546.1 ps _1133_: INVX1_155/Y -> OAI21X1_610/C
1653.7 ps _1286_: OAI21X1_610/Y -> NOR2X1_239/B
1757.8 ps _1360_: NOR2X1_239/Y -> NAND2X1_241/A
1840.1 ps _1361_: NAND2X1_241/Y -> OAI21X1_652/C
1911.8 ps _1362_: OAI21X1_652/Y -> OAI21X1_653/A
1988.2 ps _544__4_: OAI21X1_653/Y -> DFFSR_120/D
setup at destination = 82.8625
Path input pin wb_adr_i[4] to DFFSR_133/D delay 2034.56 ps
0.0 ps wb_adr_i[4]: -> INVX4_1/A
49.6 ps _241_: INVX4_1/Y -> NAND3X1_71/C
655.4 ps _251_: NAND3X1_71/Y -> NOR2X1_20/B
Quit
Edit Layout


Qflow Manager
User: vignesh
Project: qflow_spi
/home/vignesh/qflow_spi
Checklist
Preparation Okay Run Settings
Synthesis Okay Run Settings
Placement Okay Run Settings
Static Timing Analysis Okay Run Settings
Route Settings
Show graphic view:
Route layers: max
Stop flow after route success:
hold at destination = 9.68597
Path input pin wb_rst_i to DFFSR_109/S delay 159.313 ps
0.0 ps wb_rst_i: -> INVX8_11/A
30.9 ps _433_: INVX8_11/Y -> BUFX4_237/A
149.6 ps _433__bF_buf1: BUFX4_237/Y -> DFFSR_109/S
hold at destination = 9.68597
Path input pin wb_rst_i to DFFSR_98/R delay 159.982 ps
0.0 ps wb_rst_i: -> INVX8_11/A
30.9 ps _433_: INVX8_11/Y -> BUFX4_237/A
149.6 ps _433__bF_buf1: BUFX4_237/Y -> DFFSR_98/R
hold at destination = 10.3545
Path input pin wb_rst_i to DFFSR_97/R delay 159.982 ps
0.0 ps wb_rst_i: -> INVX8_11/A
30.9 ps _433_: INVX8_11/Y -> BUFX4_237/A
149.6 ps _433__bF_buf1: BUFX4_237/Y -> DFFSR_97/R
hold at destination = 10.3545
qflow exited with status 0
Current qflow status: Next project action is route
Quit
Edit Layout

Routing

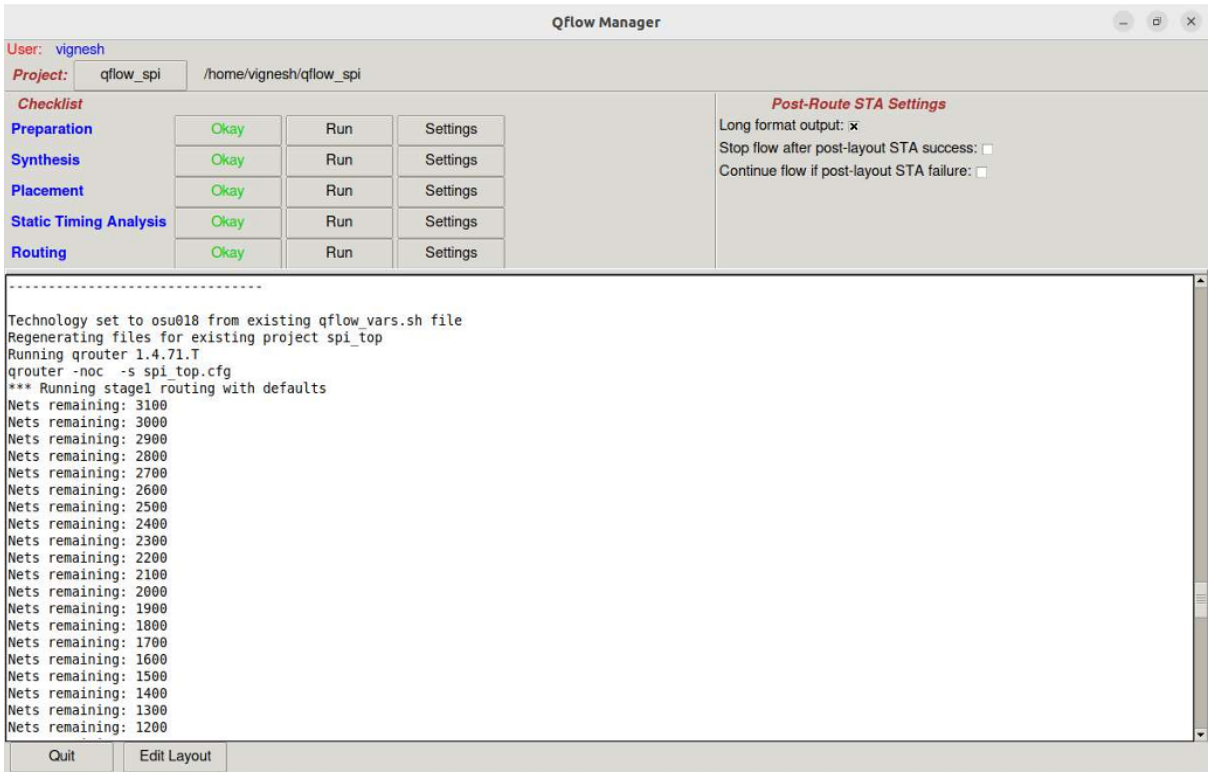
The **Routing step in Qflow GUI** is the critical stage where the physical interconnections between the placed cells in your design are established, completing the physical implementation of your ASIC layout. After the *Placement* step determines the optimal positions of all standard cells within the chip area, the Routing step ensures that every logical connection specified in your netlist is physically realized by generating metal wires between the cells on the silicon die, following the design rules and technology constraints of your chosen process. You initiate this step in the Qflow GUI by selecting your project and clicking **Routing**, which automatically calls the routing tools within Qflow to handle this complex physical task.

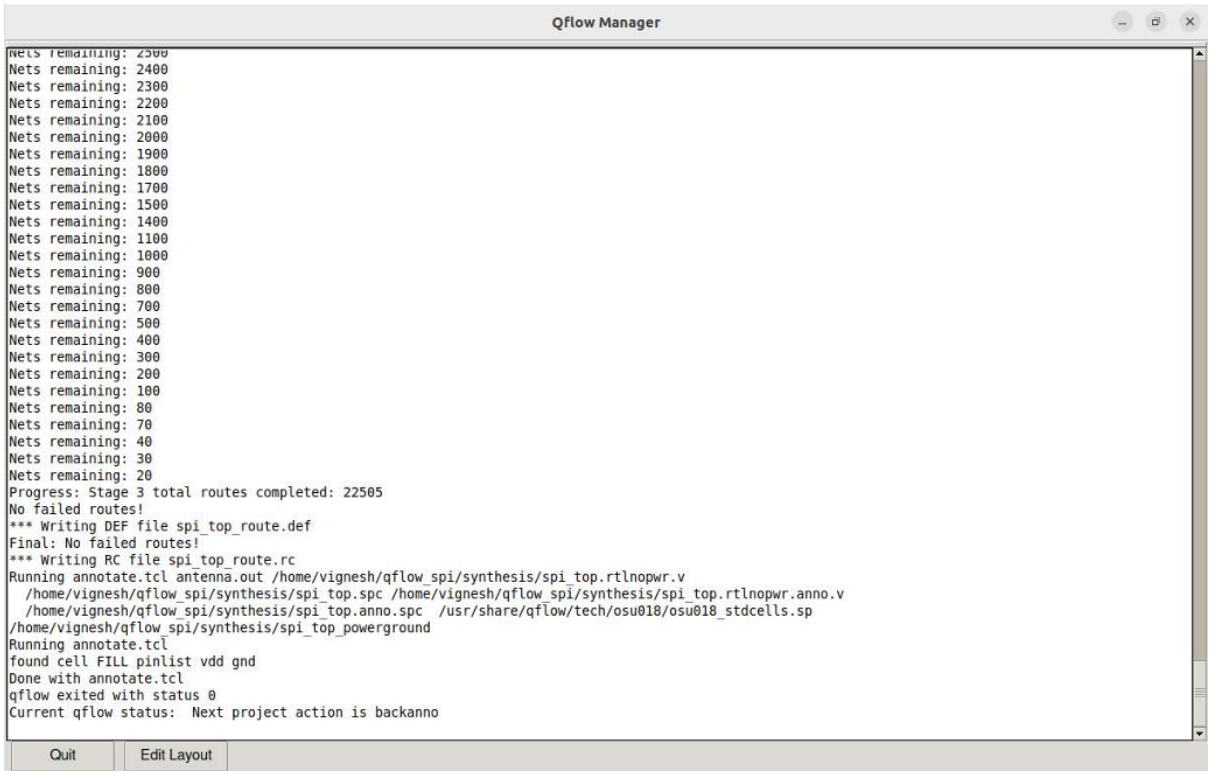
Internally, Qflow uses **Qrouter**, an open-source detail router, to perform the routing task. Qrouter takes the placed cell coordinates (.cel file) generated during placement, the standard cell library's LEF file describing the physical cell geometries and routing blockages, and the technology-specific design rules, such as minimum width, spacing, and via constraints, to perform detailed routing layer by layer. It systematically connects all nets while ensuring that there are no short circuits or design rule violations, and it handles routing congestion by iteratively refining paths when necessary to complete the routing successfully.

During routing, Qrouter uses a maze routing algorithm and iterative refinement to determine the best routing paths for each net, aiming to minimize the total wire length while respecting the available routing resources and technology constraints. The router also manages **via insertion between different metal layers**, ensuring proper connections while minimizing additional parasitic capacitance and resistance that could affect timing. If routing congestion occurs, Qrouter attempts to reroute affected nets using alternative paths, ensuring that all connections are made while maintaining a clean layout.

Once routing completes, Qflow generates several essential output files. One key output is the **routed DEF file**, which contains the complete information about the routed design, including cell placements and wire geometries, and is used for layout visualization and further analysis. Additionally, Qflow produces the **extracted SPEF or parasitic files**, which include the resistance and capacitance of the interconnects, essential for accurate post-routing static timing analysis (STA) to ensure that your design still meets timing after accounting for interconnect delays. The process also produces log files detailing routing statistics, wire lengths, via counts, and any potential warnings or issues encountered during routing.

In summary, the **Routing step in Qflow GUI connects the logical signals in your ASIC design by physically generating the metal interconnects between cells using Qrouter, respecting all design rules and minimizing wire lengths while preparing the design for timing analysis and layout generation**. This step finalizes the physical implementation of your design, setting the stage for post-routing STA, DRC checks, and GDSII generation for fabrication in your VLSI ASIC flow.


Qflow Manager
User: vignesh
Project: qflow_spi
/home/vignesh/qflow_spi
Checklist
Preparation Okay Run Settings
Synthesis Okay Run Settings
Placement Okay Run Settings
Static Timing Analysis Okay Run Settings
Routing Okay Run Settings
Post-Route STA Settings
Long format output:
Stop flow after post-layout STA success:
Continue flow if post-layout STA failure:
Technology set to osu018 from existing qflow_vars.sh file
Regenerating files for existing project spi_top
Running qrouter 1.4.71.T
qrouter -noc -s spi_top.cfg
*** Running stage1 routing with defaults
Nets remaining: 3100
Nets remaining: 3000
Nets remaining: 2900
Nets remaining: 2800
Nets remaining: 2700
Nets remaining: 2600
Nets remaining: 2500
Nets remaining: 2400
Nets remaining: 2300
Nets remaining: 2200
Nets remaining: 2100
Nets remaining: 2000
Nets remaining: 1900
Nets remaining: 1800
Nets remaining: 1700
Nets remaining: 1600
Nets remaining: 1500
Nets remaining: 1400
Nets remaining: 1300
Nets remaining: 1200
Quit
Edit Layout


Qflow Manager
Nets remaining: 2400
Nets remaining: 2300
Nets remaining: 2200
Nets remaining: 2100
Nets remaining: 2000
Nets remaining: 1900
Nets remaining: 1800
Nets remaining: 1700
Nets remaining: 1500
Nets remaining: 1400
Nets remaining: 1100
Nets remaining: 1000
Nets remaining: 900
Nets remaining: 800
Nets remaining: 700
Nets remaining: 500
Nets remaining: 400
Nets remaining: 300
Nets remaining: 200
Nets remaining: 100
Nets remaining: 80
Nets remaining: 70
Nets remaining: 40
Nets remaining: 30
Nets remaining: 20
Progress: Stage 3 total routes completed: 22505
No failed routes!
*** Writing DEF file spi_top_route.def
Final: No failed routes!
*** Writing RC file spi_top_route.rc
Running annotate.tcl antenna.out /home/vignesh/qflow_spi/synthesis/spi_top.rtlnopwr.v
/home/vignesh/qflow_spi/synthesis/spi_top.spc /home/vignesh/qflow_spi/synthesis/spi_top.rtlnopwr.anno.v
/home/vignesh/qflow_spi/synthesis/spi_top.anno.spc /usr/share/qflow/tech/osu018/osu018_stdcells.sp
/home/vignesh/qflow_spi/synthesis/spi_top_powerground
Running annotate.tcl
found cell FILL pinlist vdd gnd
Done with annotate.tcl
qflow exited with status 0
Current qflow status: Next project action is backanno
Quit
Edit Layout

Post – Route STA

The **post-Route STA step in Qflow GUI** is the stage where the **timing of your fully placed and routed ASIC design is thoroughly analysed to ensure it meets the specified clock frequency and timing constraints** after considering the actual physical interconnections and parasitic effects introduced during routing. Before routing, STA typically uses estimated wire delays; however, the post-route STA leverages **accurate parasitic extraction data**, including the actual resistances and capacitances of the routed wires and vias, providing a realistic evaluation of your design's timing performance. You initiate this step in Qflow GUI by clicking **Analysis** after completing the Routing step.

Internally, Qflow uses **vesta**, an open-source static timing analysis tool, to perform post-route STA. It reads your **routed netlist (.blif or .v)**, the **standard cell timing library (.lib)** describing delays of your cells, and the **parasitic files (.rc or .spef)** generated during routing, which contain detailed resistance and capacitance information for the wires in your design. Vesta uses this data to calculate the **arrival and required times of signals across all timing paths in your design**, checking whether signals arrive at their destinations within the setup and hold timing requirements relative to the defined clock period.

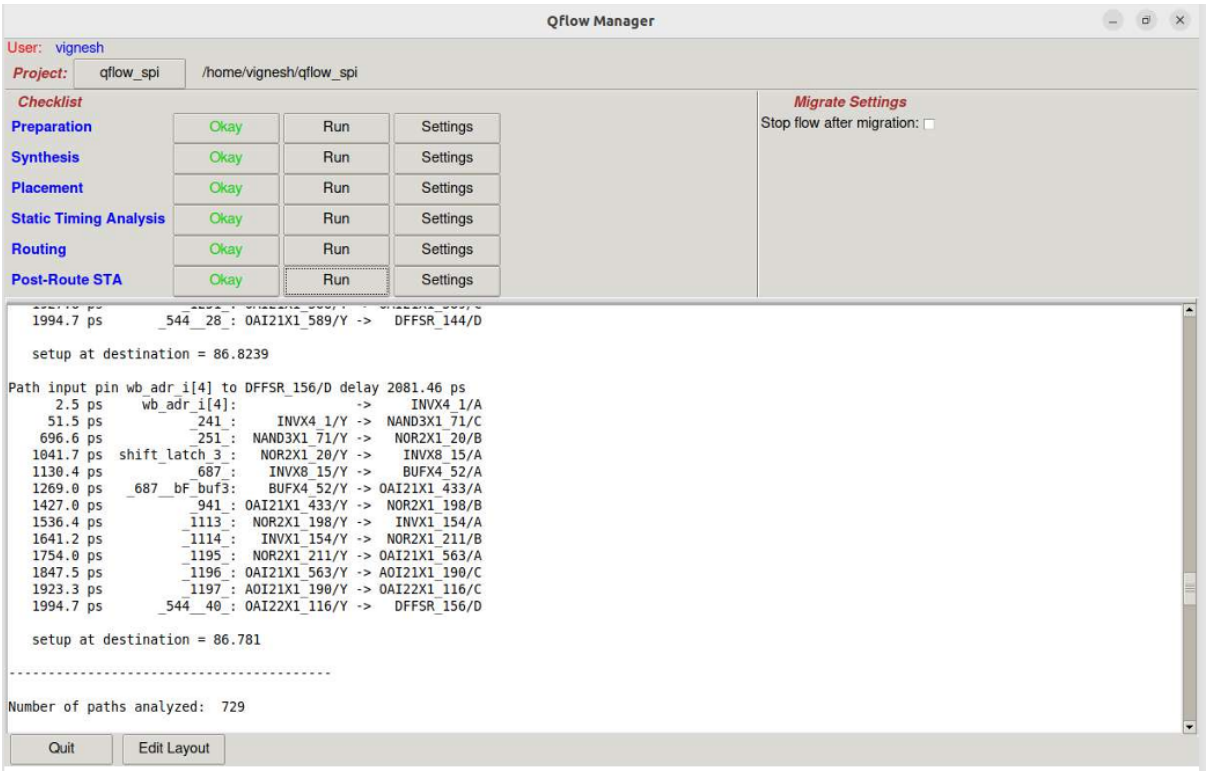
During this step, vesta systematically analyzes **all critical timing paths** in your design, including paths between flip-flops (register-to-register), from input ports to registers, and from registers to output ports. It computes **slack values** for each path, where slack is the difference between the required time and the actual arrival time of a signal. A positive slack indicates the design meets timing

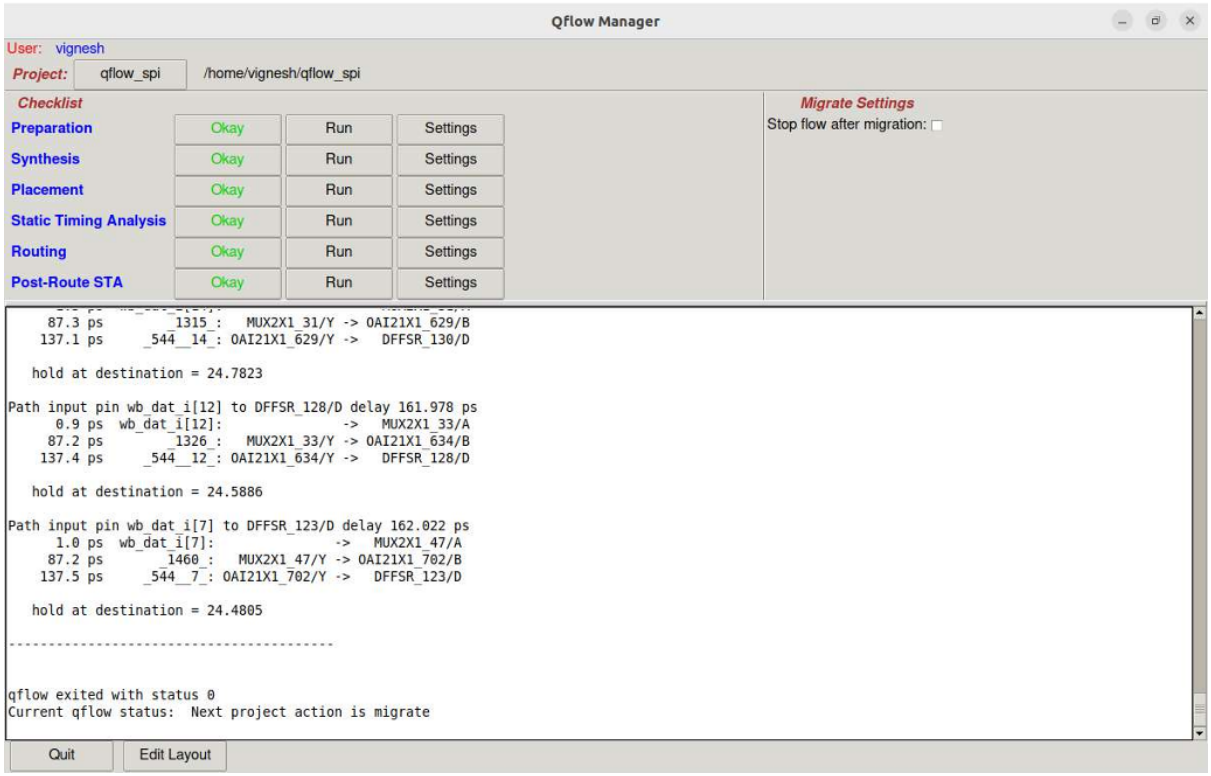
on that path, while a negative slack indicates a timing violation. This process identifies the **critical path**, which is the path with the least (or most negative) slack, effectively determining the maximum operational frequency your design can reliably support.

Post-route STA is crucial because the addition of real parasitics from routing often causes delays to increase, which may lead to timing violations even if pre-route STA passed successfully. If timing violations are reported, you may need to revise your clock constraints, optimize your RTL to shorten critical paths, or adjust placement and routing strategies to reduce wire lengths in critical areas to meet the desired clock frequency.

After completing the analysis, Qflow generates **detailed timing reports** that include the maximum delays, slack values for all paths, identification of the critical path, and a summary indicating whether your design meets the timing requirements. These reports are essential for final sign-off before layout generation, ensuring your design will operate correctly at the targeted frequency when fabricated.

In summary, the **post-Route STA step in Qflow GUI verifies whether your fully placed and routed ASIC design meets its timing constraints by analysing the true delays using extracted parasitics with the vesta tool**, providing confidence in your design's performance before tape-out. This step is critical in ensuring your ASIC will function as intended in silicon, making it an essential final timing validation stage in your VLSI digital design flow.





Migration

The **Migration step in Qflow GUI** is the stage where your fully placed and routed ASIC design is prepared for **final layout generation, export, and tape-out** by converting your routed design into a standard layout format suitable for further physical verification and fabrication. After post-route STA confirms that your design meets timing requirements, the migration step ensures that your design is **translated into a physical layout format such as GDSII**, which is the industry standard used by fabrication facilities for chip manufacturing. You execute this step in the Qflow GUI by selecting your project and clicking **Migration**, which automates the conversion process using the integrated layout generation tools.

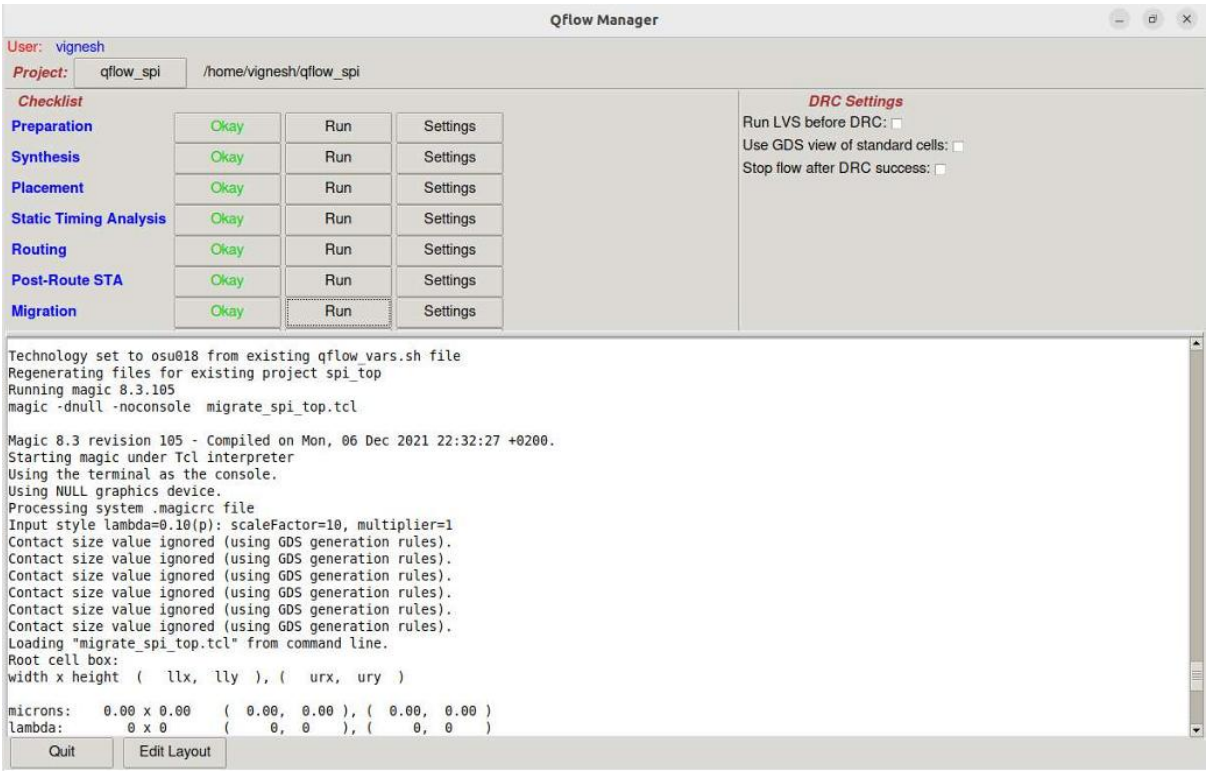
Internally, during the Migration step, **Qflow uses magic, an open-source layout tool**, to convert your **DEF (Design Exchange Format) files generated after routing into a GDSII layout**. The DEF file contains detailed information about the placement of cells and the routing of interconnects, while the LEF file provides abstract physical definitions of standard cells. Magic interprets these files along with your technology-specific information to reconstruct your ASIC's complete physical layout, including the exact geometry of the metal layers, vias, and cell placements, ensuring that the generated layout accurately reflects your design's logical and physical implementation.

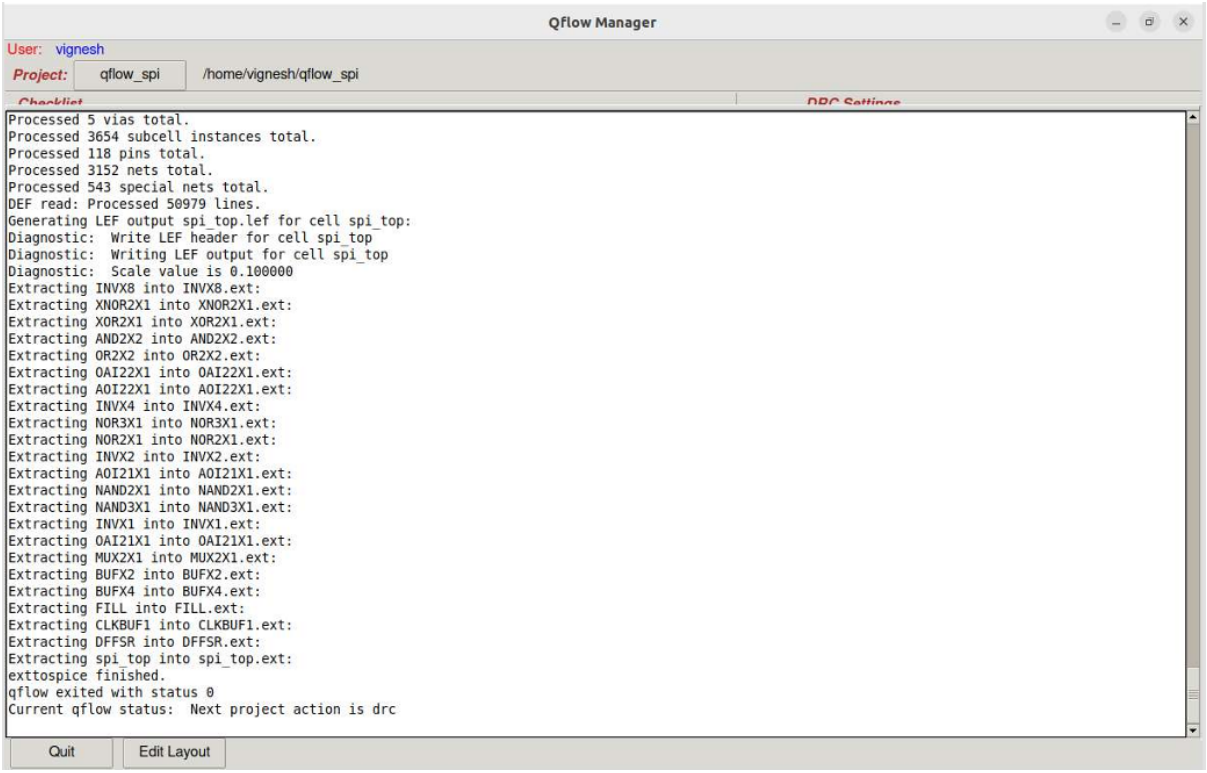
The Migration step also performs **layout clean-up tasks**, including the alignment of cells to the manufacturing grid, removal of minor geometry inconsistencies, and fixing minor design rule check (DRC) issues that can be automatically resolved without modifying the design's functionality. This ensures that your layout adheres to the manufacturing rules specified by the technology process, which is crucial for a successful tape-out and fabrication.

During this process, Magic also allows you to **visualize your ASIC layout**, providing a way to inspect the placement of cells, the routing of interconnects across different metal layers, and the integration of IO pads and macros if present in your design. This visual inspection helps you verify that the layout generated corresponds accurately to your expectations before final export.

Upon completion, the Migration step generates the **GDSII file of your design**, which serves as the final layout file for your ASIC. This GDSII file can then be used for **physical verification processes**, including Design Rule Checking (DRC), Layout Versus Schematic (LVS) checks, and further parasitic extraction for sign-off timing analysis. It also enables the submission of your design to the foundry for fabrication, making it a critical step in your VLSI flow.

In summary, the **Migration step in Qflow GUI translates your fully routed ASIC design into the industry-standard GDSII layout format using Magic, performing layout clean-up, alignment, and visualization to prepare your design for final physical verification and fabrication**. This step ensures your logically correct and timing-verified design is now fully ready for tape-out, completing your digital RTL-to-GDSII flow systematically and robustly.





After this we can run LVS before DRC.

Layout Versus Schematic (LVS)

The **LVS (Layout Versus Schematic) step in Qflow GUI** is a critical stage where you **verify that your final physical layout (post-routing) exactly matches your intended logical design (schematic or netlist)** before proceeding to tape-out and fabrication. This ensures that what you have physically implemented on silicon through placement and routing truly represents your RTL's logical functionality without missing connections, shorts, or unintended modifications during the physical design process. In Qflow, this step helps catch critical issues that could otherwise lead to non-functional chips or unexpected behavior after fabrication.

In the Qflow GUI, you perform this step by selecting your project and clicking the **LVS** option, allowing Qflow to automate the LVS process using its integrated toolchain. Internally, Qflow typically uses **Netgen**, an open-source LVS tool, to perform this comparison. Netgen compares the **netlist extracted from your final layout (using Magic) with your original synthesized netlist**, which was generated during the synthesis stage using Yosys. The tool cross-checks every net, node, and component, verifying that all transistors, logic gates, and their connections in your layout correspond exactly to those in your schematic netlist, down to the port connections and hierarchical structure. During the LVS step, the extracted netlist from Magic includes the details of the layout's geometry converted into electrical connectivity information, such as how metal layers, vias, and transistors are interconnected. Netgen reads this extracted netlist and the synthesized netlist, then systematically checks for discrepancies such as:

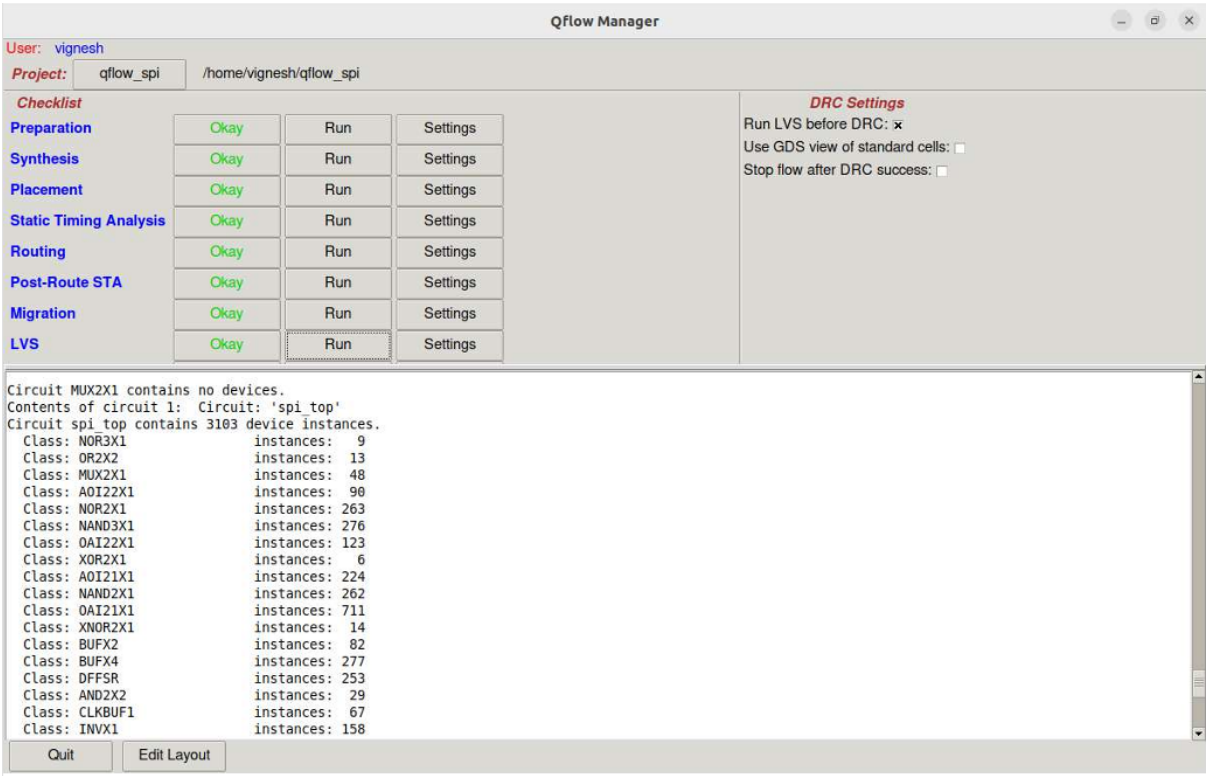
- **Missing connections (opens)**, indicating incomplete routing.
- **Unintended connections (shorts)**, indicating possible design rule violations or routing errors.

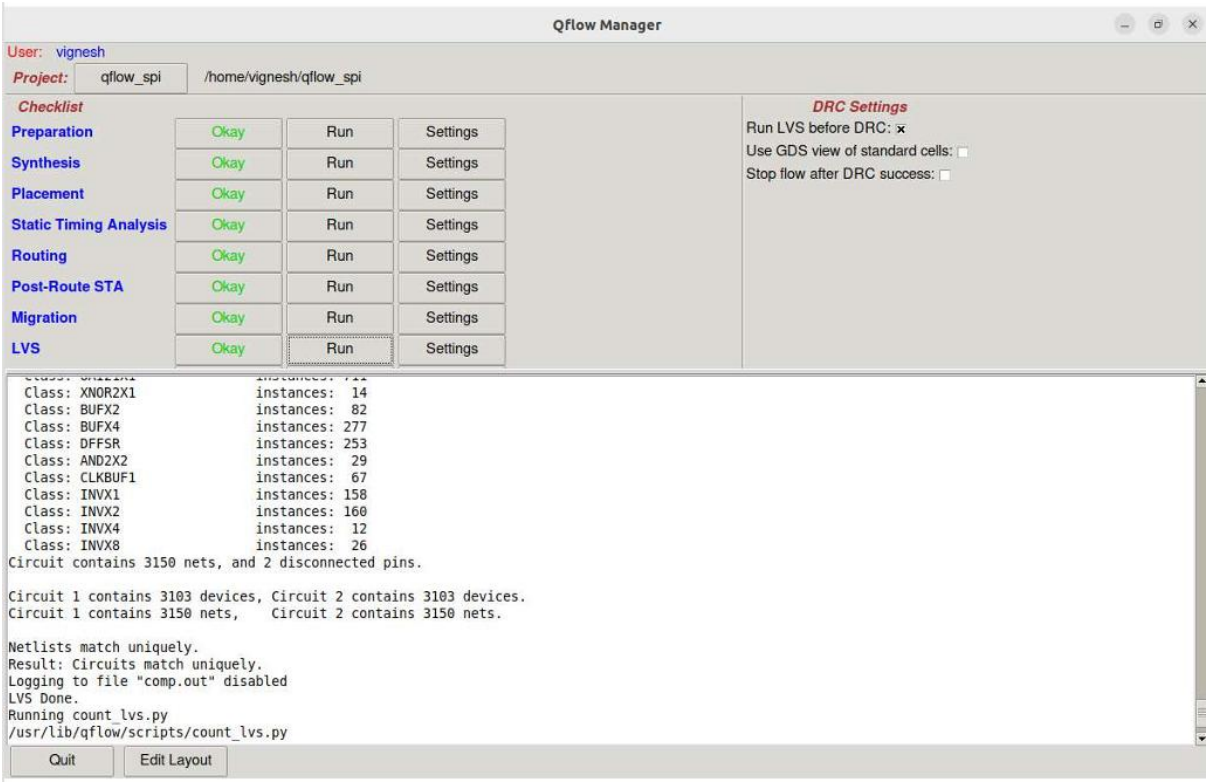
- **Incorrect pin or port connections**, where a signal might be connected to the wrong input or output.
- **Device parameter mismatches**, such as transistor sizes differing between layout and schematic.

Once the LVS comparison completes, Netgen generates a detailed **LVS report** showing whether the layout matches the schematic ("circuits match") or lists the specific discrepancies encountered if mismatches are found. It provides a clear summary of nets and devices present in both netlists and pinpoints the exact areas requiring debugging. If errors occur, you can return to the Magic layout, adjust your routing or layout geometry, or correct the logical netlist if necessary to align the layout with your intended design.

The LVS step is crucial because even if your design passes functional simulation, synthesis, placement, routing, and post-route STA, any mismatch between the final layout and the intended design can result in a non-functional chip. Therefore, **LVS ensures correctness before committing the design to silicon, acting as the final verification step in your digital ASIC design flow in Qflow**.

In summary, the **LVS step in Qflow GUI uses Netgen to compare your extracted layout netlist against your synthesized netlist, verifying that your final layout precisely implements your logical design without unintended opens, shorts, or connectivity errors, ensuring your design is ready for tape-out and fabrication**. This step guarantees that your RTL-to-GDSII flow is not only timing-verified but also functionally correct in its physical implementation, securing your VLSI design pipeline's robustness and reliability.





Design Rule Check (DRC)

The **DRC (Design Rule Check) step in Qflow GUI** is the stage where your completed ASIC layout is checked against the **physical manufacturing constraints defined by the semiconductor foundry**, ensuring that your design can be fabricated without errors and will function reliably in silicon. Design rules specify the minimum and maximum dimensions, spacings, and overlaps for various geometric structures in the layout, such as metal layers, vias, and transistor regions, to ensure manufacturability and yield. The DRC step is essential because even if your design is logically correct and passes LVS and timing checks, violations of these geometric constraints can result in fabrication defects, leading to chip failures or low yield.

In Qflow, you perform this step by selecting your project and clicking **DRC**, which automates the process using the integrated **Magic VLSI layout tool**. Magic reads your **final layout file (usually in .mag format) generated during the Migration step** and checks the layout geometries against the design rules provided for your chosen technology node (e.g., 180nm or 130nm). These rules include specifications like minimum metal widths, minimum spacing between metal tracks, enclosure rules for vias, and well-to-active region spacing, among others. Magic applies these rules systematically across the entire layout, identifying any places where the design violates the technology's manufacturing constraints.

During DRC, Magic highlights **each violation with precise geometric coordinates and layer information**, allowing you to locate the exact areas in the layout where corrections are needed. Common issues found during DRC include:

- Metal traces being too close to each other (spacing violations).

- Metal widths being too narrow.
- Vias not having the required enclosure from the metal layers.
- Overlaps between incompatible layers.
- Minimum area violations on certain layers.

After completing the check, Magic generates a **detailed DRC report**, summarizing the total number of violations, the types of violations, and their exact locations in the layout. You can visualize these violations directly within the Magic GUI, where errors are highlighted for interactive debugging. If violations are found, you need to modify the layout in Magic, often by adjusting the placement of cells, rerouting wires, resizing geometries, or fixing via enclosures to resolve the issues while ensuring that the functionality and timing of your design remain intact.

The DRC step is crucial because semiconductor foundries will typically refuse to fabricate a design that does not pass DRC, as it indicates a high risk of yield loss or functional failures. Passing DRC gives you confidence that your layout is clean and manufacturable, ensuring that your design can move forward to tape-out and fabrication without physical implementation issues.

In summary, the **DRC step in Qflow GUI verifies that your ASIC layout adheres to the manufacturing constraints defined by your target technology using Magic, identifying and helping you correct geometric violations to ensure your design is manufacturable and reliable in silicon**. This step finalizes your physical verification flow, completing your RTL-to-GDSII process robustly before tape-out and ensuring your design is ready for fabrication.

Graphic Data System (GDS)

The **GDS (GDSII Generation) step in Qflow GUI** is the final stage in your RTL-to-GDSII ASIC design flow, where your fully verified, DRC-clean, LVS-matched layout is exported into the **GDSII file format**, the industry-standard format used by semiconductor foundries for chip fabrication. GDSII (Graphic Data System II) is a binary format that stores the geometric data of the layout, including the positions and shapes of all polygons, layers, vias, and text labels, ensuring the physical design can be precisely reproduced during the photolithography steps in manufacturing.

In Qflow GUI, you perform this step by selecting your project and clicking **GDS**, which automates the generation process using **Magic**, the open-source layout editor integrated within Qflow. Magic reads your **final layout (.mag file) generated and cleaned during the Migration and DRC steps** and converts it into a **.gds** file while preserving the hierarchical structure, layer information, and precise geometries necessary for fabrication. During this process, Magic also applies the necessary layer mapping, ensuring that the internal layer names used during design correspond to the foundry's expected layer numbers in the GDSII file.

The GDS generation step ensures that **all design layers, including active areas, polysilicon, metal layers, vias, and wells, are correctly represented in the exported file**, maintaining the physical fidelity of your design. It also includes the pad frame and IO cells, ensuring that the complete chip structure is exported for packaging and bonding during fabrication. Additionally, this step can include the generation of fill patterns or dummy layers if your technology requires them to maintain planarity during fabrication.

After the GDS export is complete, Qflow generates the **.gds** file, which serves as the **final deliverable to the semiconductor foundry** for fabrication. This file can also be viewed using GDS viewers or re-imported into Magic or other layout tools to verify the correctness of the export visually. The size of the GDSII file typically depends on the complexity and area of your ASIC design, and it contains all the hierarchical cell structures used in your layout.

The GDS step is crucial because **this file is what the foundry will use to create the photomasks required for fabricating your chip**, making it essential that it accurately reflects your intended design without errors. Any discrepancy at this stage could lead to silicon failures or non-functional chips, emphasizing the importance of ensuring that all previous steps (DRC, LVS, STA) are clean before performing the GDS export.

In summary, the **GDS step in Qflow GUI converts your final, verified ASIC layout into the industry-standard GDSII format using Magic, ensuring your design is ready for fabrication while preserving the physical integrity and hierarchical structure of your layout for accurate reproduction on silicon**. This step marks the culmination of your RTL-to-GDSII design flow, enabling you to confidently proceed to tape-out and manufacturing with your ASIC design.

Clean Up

The **Clean Up step in QFlow GUI** is the final housekeeping stage in your RTL-to-GDSII ASIC design flow, where temporary files and intermediate artifacts generated during each stage of the design process are systematically removed to free up disk space and organize your project directory for archival, sharing, or future reference. Throughout the Qflow pipeline, multiple intermediate files are generated during preparation, synthesis, placement, routing, analysis, and layout generation. These include .blif files, temporary Verilog files, intermediate DEF files, placement logs, routing trial outputs, parasitic extraction data, temporary timing reports, and simulation artifacts, many of which are no longer needed after successful completion of your flow and verification.

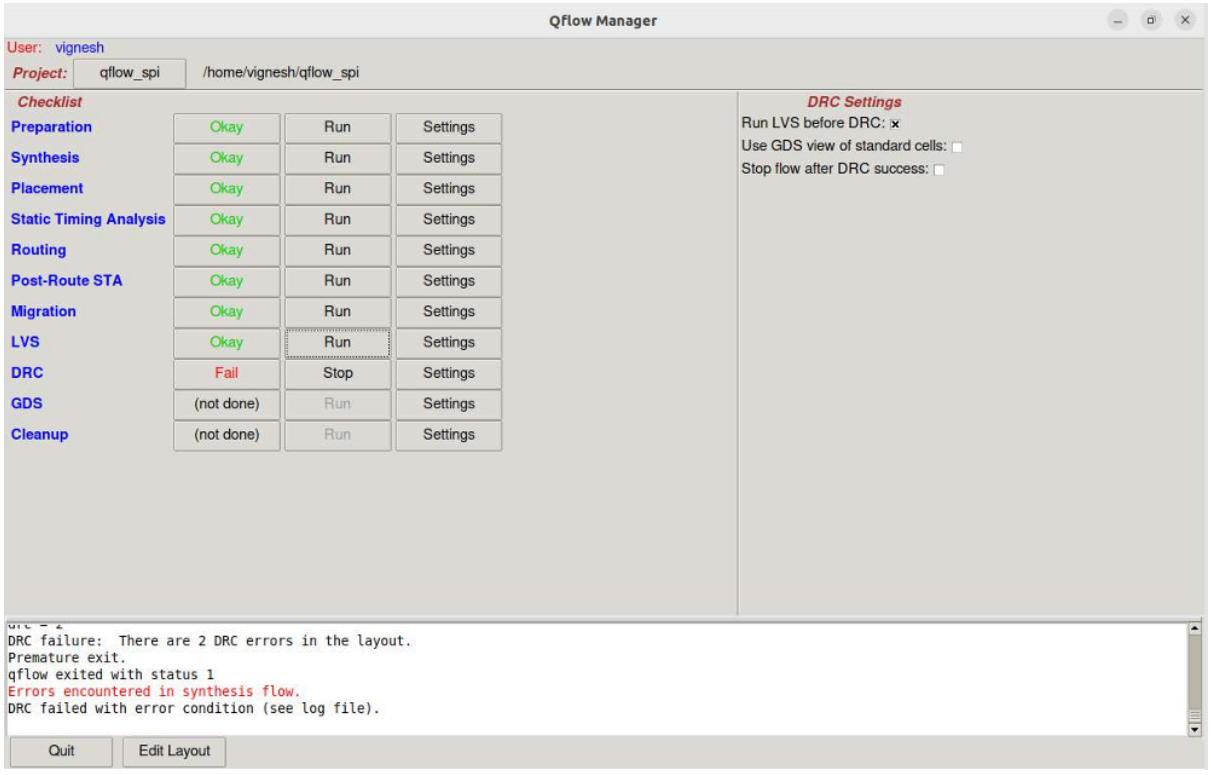
You initiate this step in the QFlow GUI by selecting your project and clicking **Clean Up**, which automatically triggers a scripted process to identify and remove these unnecessary files while carefully preserving the essential project deliverables. The essential files typically retained include your **final GDSII layout (.gds), the final routed DEF file, critical synthesis and timing reports for documentation, and your original RTL source files**. By removing unused or redundant data, the Clean Up step helps maintain a tidy project environment, making it easier to archive your project or transfer it for tape-out submission.

An important role of the Clean Up step is to **recover significant disk space**, which is especially valuable when working on large ASIC designs where intermediate files can occupy gigabytes of storage. This allows your working environment to remain efficient and prevents clutter that can cause confusion in your design directory structure. It also ensures that any subsequent runs of Qflow on the same project directory will not pick up stale files from previous runs, reducing the risk of inconsistencies or conflicts during re-synthesis, re-routing, or re-analysis.

Additionally, the Clean Up step in Qflow acts as a preparation for **project closure and handover**, ensuring that only the files necessary for documentation, tape-out, or further post-layout analysis (such as advanced sign-off tools) are retained. This helps enforce good design practices, emphasizing the importance of maintaining a clean, documented, and reproducible design flow, which is essential in professional VLSI workflows where project handoffs are common.

In summary, the **Clean Up step in QFlow GUI efficiently removes unnecessary temporary and intermediate files from your project directory while preserving essential design outputs like your GDSII, RTL, and critical reports, helping maintain an organized, storage-efficient, and clean environment as you conclude your RTL-to-GDSII design flow**. This final step ensures your project is ready for archiving, sharing, or submission to fabrication while embodying good engineering and project management discipline.

Final Output



Hence the Serial Peripheral Interface is implemented using QFlow.

A Project Report

by

Yemireddy Vignesh Reddy