PHYSICAL DESIGN IMPLEMENTATION OF SPI (SERIAL PERIPHERAL INTERFACE) USING QFLOW

STUDENT NAME: JAGRUTHI KARAMALA

BATCH NAME: B2B PDI 01

REGISTRATION NUMBER: 22BEC7269

UNIVERSITY NAME: VIT AP UNIVERSITY

COURSE VENDOR: MAVEN SILICON

TOOL USED: QFLOW

EMAIL ID: jagruthi.22bec7269@vitapstudent.ac.in

INTRODUCTION

The demand for open-source digital design tools has grown rapidly in recent years. Among these tools, Qflow has emerged as a reliable and efficient digital synthesis and layout flow for implementing digital circuits using standard cells. This project focuses on the Physical Design Implementation of a Serial Peripheral Interface (SPI) using the Qflow toolchain, targeting the osu018 open-source standard cell library.

The goal of this project is to take a Verilog RTL description of an SPI master module and convert it all the way into a manufacturable GDSII layout using Qflow. This journey from RTL to GDS involves multiple critical stages including synthesis, placement, routing, DRC (Design Rule Check), LVS (Layout vs Schematic), and final GDS generation.

In the semiconductor industry, the physical design flow forms the backbone of backend VLSI design. It ensures that the logical design is transformed into a layout that meets all timing, area, power, and manufacturing constraints. By using Qflow, an open-source digital backend flow, this project not only simulates a real-world ASIC design process but also highlights the feasibility of using open-source tools for academic and prototyping purposes.

This report details each step of the implementation process including:

- Overview of SPI protocol and its significance
- Verilog-based RTL coding
- Tool setup and invocation
- Synthesis of RTL to gate-level netlist
- Cell placement and wire routing
- Static Timing Analysis (STA)
- Design Rule Checking (DRC)
- Layout vs. Schematic comparison (LVS)
- Export of final GDSII file for fabrication

Serial Peripheral Interface (SPI)

The Serial Peripheral Interface (SPI) is a synchronous serial communication protocol developed by Motorola. It is widely used in embedded systems to provide high-speed communication between microcontrollers and peripheral devices such as sensors, memory chips, ADCs, and DACs.

SPI operates in a master-slave configuration. Only one master is allowed in the bus system, but multiple slaves can be connected. Communication is full-duplex, meaning data can be transmitted and received simultaneously. SPI uses four primary signals:

- MOSI (Master Out Slave In): Line for data sent from master to slave.
- MISO (Master In Slave Out): Line for data sent from slave to master.
- SCLK (Serial Clock): Clock signal generated by the master to synchronize communication.
- SS or CS (Slave Select / Chip Select): Active-low signal to select a specific slave device.

Only the slave whose SS line is pulled low will respond to the master. Other slaves remain inactive.

SPI Modes

SPI has four modes of operation based on two clock parameters:

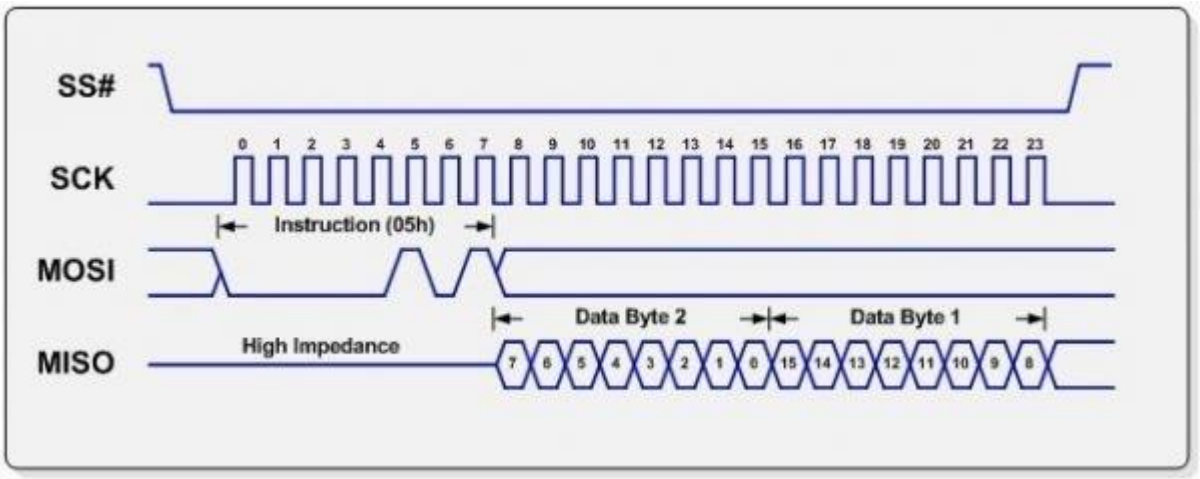
- CPOL (Clock Polarity): Determines the idle state of the clock.
- CPHA (Clock Phase): Determines the clock edge (rising/falling) on which data is captured.
- SPI Timing

In SPI Mode 0:

- Data is stable on the rising edge of SCLK.
- Data must be set up before the rising edge.
- Transmission begins when the SS signal goes low.

Each transmission consists of 8 bits (1 byte) or more depending on the implementation. Typically, a shift register is used to shift bits out (MOSI) or in (MISO) one bit at a time with each clock cycle.

TIMING DIAGRAM



INVOKE THE TOOL

FROM THE PROJECT DIRECTORY (THAT IS, "CD" TO THE PROJECT DIRECTORY), RUN: QFLOW GUI

USER@JAGRUTHI:~/QFLOW_SPI$ QFLOW GUI

AT THE TOP IS THE "PROJECT" NAME, WHICH SHOWS THE WORKING DIRECTORY INCLUDES THE PROJECT FILES. THE WORKING DIRECTORY NAME IS CONSIDERED THE NAME OF THE PROJECT. THE FULL PATH TO THE DIRECTORY IS SHOWN TO THE RIGHT OF THE BUTTON.

"CHECKLIST" AND “SETTINGS” ARE TWO SECTIONS PLACED ON THE LEFT AND RIGHT SIDE UNDER THE PROJECT. THE CHECKLIST IS A LIST OF STEPS PROCEEDING THROUGH THE SYNTHESIS FLOW, STARTING FROM PREPARING THE WORKSPACE ALL THE WAY TO GENERATING A GDS OUTPUT FILE AND CLEANING UP UNNEEDED WORKING FILES.

BESIDE EACH STEP IN THE CHECKLIST, THREE BUTTONS ARE PRESENT. THE LEFTMOST BUTTON SHOWS THE STATUS OF THE STEP, FROM "(NOT DONE)" TO "OKAY" TO "FAIL", DEPENDING ON WHETHER OR NOT THE STEP HAS BEEN COMPLETED, OR SHOWING THE RESULT OF THE STEP. CLICKING ON THIS BUTTON GIVES FURTHER DETAILS BY DISPLAYING THE LOG FILE FOR THAT STEP, WHICH IS MOSTLY USEFUL ONLY IF THE STEP FAILED.

THE MIDDLE BUTTON IS THE EXECUTION BUTTON AND EITHER SAYS "RUN" TO RUN THE SYNTHESIS FLOW STEP, OR "STOP" IF THE STEP IS RUNNING. THE ITEM THAT IS NEXT TO RUN IN THE FLOW WILL HAVE "RUN" IN BOLDFACE. THOSE STEPS AHEAD WHICH CANNOT BE RUN UNTIL OTHER STEPS HAVE RUN FIRST WILL BE DISABLED (GRAYED OUT). STEPS WHICH HAVE ALREADY BEEN RUN DISPLAY "RUN" IN NORMAL TYPE. IT IS ALWAYS POSSIBLE TO GO BACK AND RE-RUN A STEP THAT HAS BEEN RUN PREVIOUSLY. THE BUTTON ON THE RIGHT IS THE "SETTINGS" BUTTON. THIS CONTROLS WHAT IS DISPLAYED IN THE SETTINGS WINDOW ON THE RIGHT, WHICH IS DIFFERENT FOR EACH STEP. NORMALLY, THE DISPLAY IN THE SETTINGS WINDOW CORRESPONDS TO THE NEXT STEP IN THE FLOW TO RUN. HOWEVER, BY CLICKING ON THE BUTTON, YOU CAN REVIEW THE SETTINGS FOR ANY STEP OF THE PROCESS.

HE SETTINGS WINDOW IS, AS MENTIONED ABOVE, DIFFERENT FOR EACH STEP. EACH WINDOW SHOWS THE SETTINGS THAT THE USER CAN CONTROL FOR THAT STEP. SETTINGS CAN BE CHANGED AT ANY TIME, ALTHOUGH SETTINGS CHANGED FOR A STEP AFTER THE STEP HAS BEEN RUN WILL NOT APPLY UNTIL THAT STEP IS RUN AGAIN.

TOOL AND SETUP ENVIRONMENT

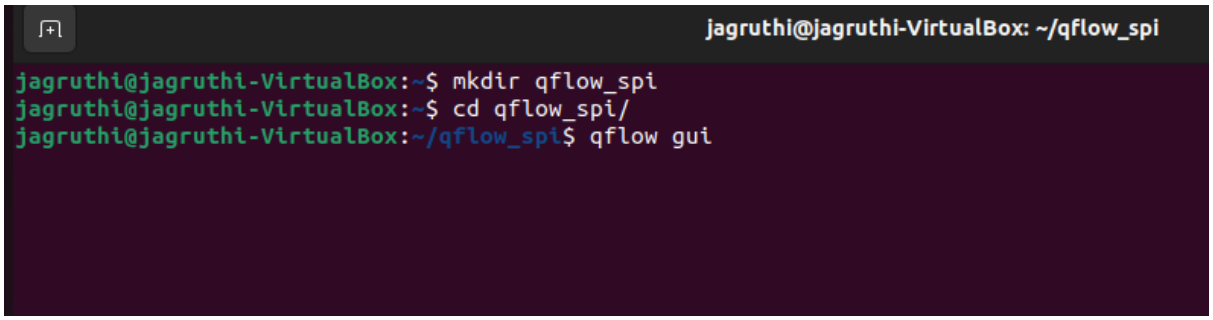
BEFORE STARTING THE FLOW, ENSURE THE FOLLOWING TOOLS ARE INSTALLED:

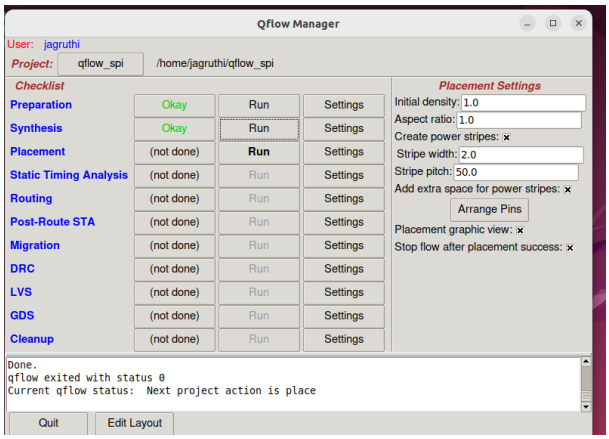
- QFLOW
- YOSYS (FOR SYNTHESIS)
- GRAYWOLF (FOR PLACEMENT)
- QROUTER (FOR ROUTING)
- MAGIC (FOR DRC/LVS AND LAYOUT VIEWING)
- NETGEN (FOR LVS)

INSTALL USING:

SUDO APT-GET INSTALL QFLOW YOSYS GRAYWOLF QROUTER MAGIC NETGEN

ALSO, SET UP YOUR .BASHRC OR .ZSHRC FILE WITH PATHS TO QFLOW TOOLS FOR SMOOTH INVOCATION.





SYNTHESIS

SYNTHESIS IS THE PROCESS OF TRANSFORMING HIGH-LEVEL DIGITAL DESIGNS WRITTEN IN A HARDWARE DESCRIPTION LANGUAGE (HDL), SUCH AS VERILOG, INTO A GATE-LEVEL NETLIST COMPOSED OF STANDARD CELLS FROM A TECHNOLOGY LIBRARY. THIS IS THE FIRST MAJOR STEP IN THE PHYSICAL DESIGN FLOW AND ACTS AS THE BRIDGE BETWEEN FRONT-END DESIGN AND BACK-END LAYOUT.

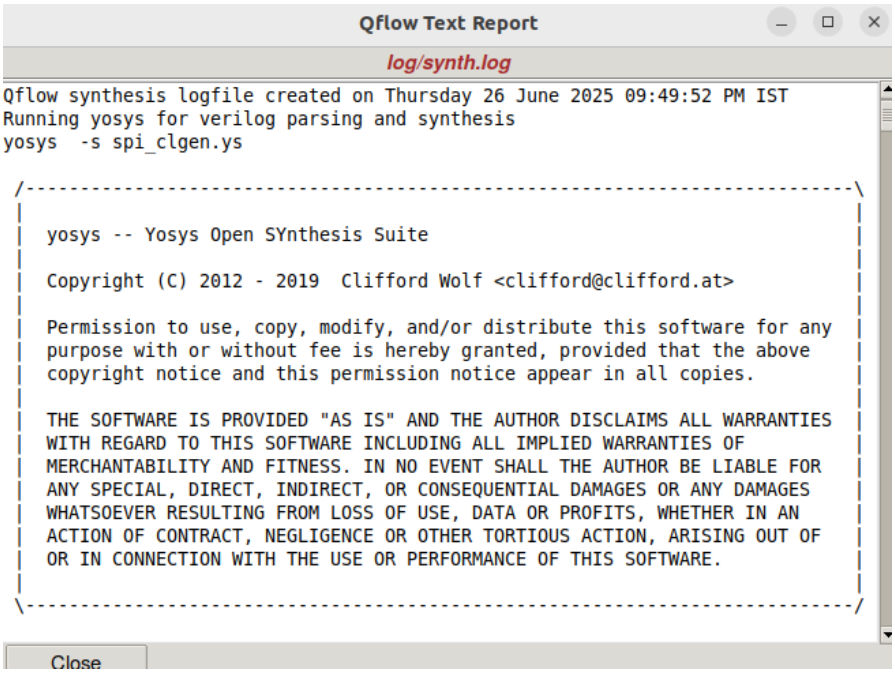
THE GOAL OF SYNTHESIS IS TO:

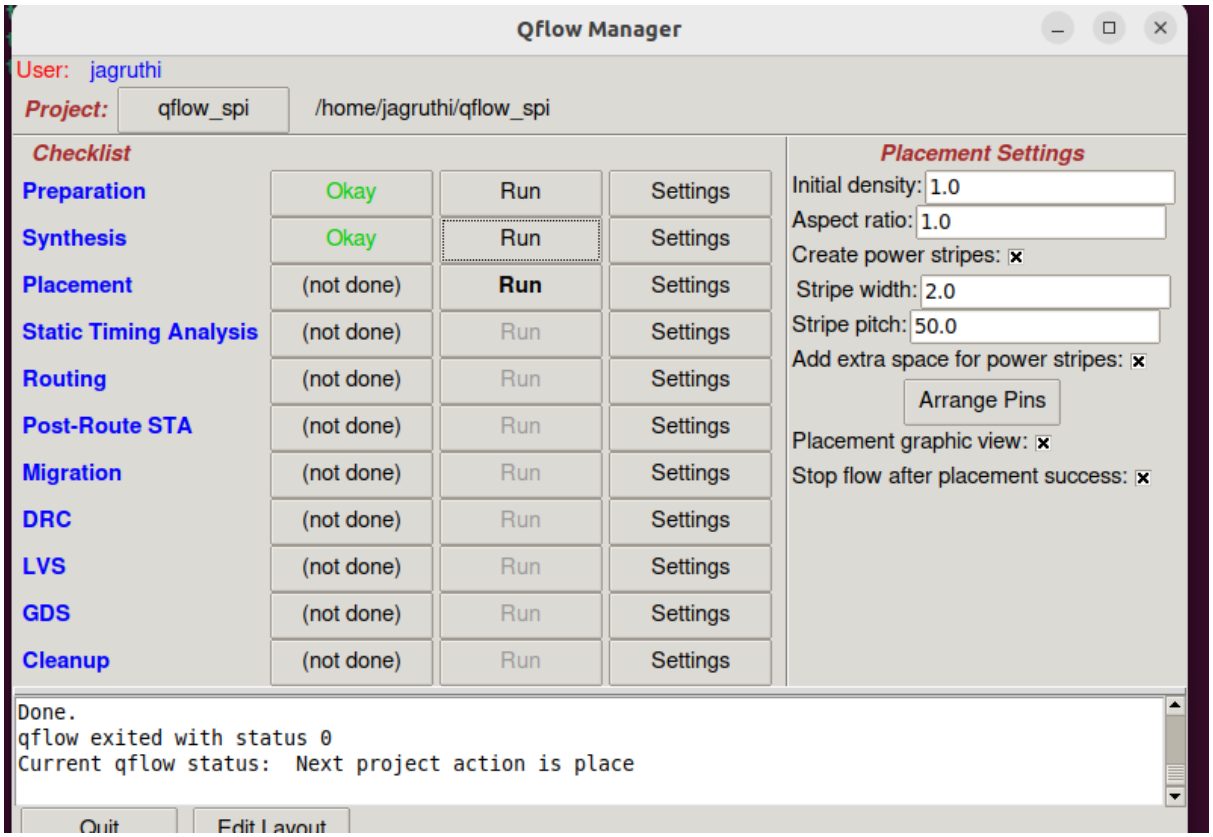
- TRANSLATE BEHAVIORAL RTL CODE INTO LOGIC GATES.
- OPTIMIZE THE DESIGN FOR AREA, TIMING, AND POWER.
- MAP THE DESIGN TO ACTUAL STANDARD CELLS FROM A TECHNOLOGY LIBRARY.

IN QFLOW, YOSYS IS USED FOR RTL SYNTHESIS. THIS CONVERTS THE VERILOG CODE INTO A GATE-LEVEL NETLIST USING STANDARD CELLS (OSU035 OR OSU018 LIBRARY). KEY STEPS:

- RTL PARSING
- OPTIMIZATION
- TECHNOLOGY MAPPING

THE OUTPUT IS A SYNTHESIZED .BLIF FILE AND .V GATE-LEVEL NETLIST. THIS STEP ALSO CHECKS FOR SYNTAX CORRECTNESS AND TIMING CONSTRAINTS USING .SDC FILE (IF PROVIDED).


Qflow Text Report
log/synth.log
Qflow synthesis logfile created on Thursday 26 June 2025 09:49:52 PM IST
Running yosys for verilog parsing and synthesis
yosys -s spi_clgen.ys
yosys -- Yosys Open SYnthesis Suite
Copyright (C) 2012 - 2019 Clifford Wolf <clifford@clifford.at>
Permission to use, copy, modify, and/or distribute this software for any
purpose with or without fee is hereby granted, provided that the above
copyright notice and this permission notice appear in all copies.
THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
Close


Qflow Manager
User: jagruthi
Project: qflow_spi
/home/jagruthi/qflow_spi
Checklist
Preparation Okay Run Settings
Synthesis Okay Run Settings
Placement (not done) Run Settings
Static Timing Analysis (not done) Run Settings
Routing (not done) Run 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
Placement Settings
Initial density: 1.0
Aspect ratio: 1.0
Create power stripes:
Stripe width: 2.0
Stripe pitch: 50.0
Add extra space for power stripes:
Arrange Pins
Placement graphic view:
Stop flow after placement success:
Done.
qflow exited with status 0
Current qflow status: Next project action is place
Quit
Edit Layout

```
Gate counts by drive strength:

        "" gates        In: 253         Out: 253        +0
        "1" gates       In: 2251        Out: 2251       +0
        "2" gates       In: 284         Out: 284        +0
        "4" gates       In: 289         Out: 289        +0
        "8" gates       In: 26          Out: 26         +0

Number of gates changed: 0
gates resized: 0
```

```
=== spi_top ===

Number of wires:                2188
Number of wire bits:            2806
Number of public wires:         2188
Number of public wire bits:     2806
Number of memories:                0
Number of memory bits:             0
Number of processes:               0
Number of cells:                2746
AND2X2                            29
AOI21X1                          224
AOI22X1                           90
BUFX2                             69
DFFSR                            253
INVX1                            356
MUX2X1                            48
NAND2X1                          262
NAND3X1                          276
NOR2X1                           263
NOR3X1                             9
OAI21X1                          711
OAI22X1                          123
OR2X2                             13
XNOR2X1                           14
XOR2X1                             6
```

Placement

Placement is a critical step in the physical design flow where the synthesized logic gates (standard cells) are assigned fixed locations on the chip floorplan. The goal is to place these cells in such a way that they minimize wirelength, reduce delay, and avoid overlap, while satisfying routing and timing constraints.

In Qflow, Graywolf is the tool used to perform automated standard-cell placement. After synthesis using Yosys, Graywolf arranges the logic gates spatially based on their connectivity described in the netlist.

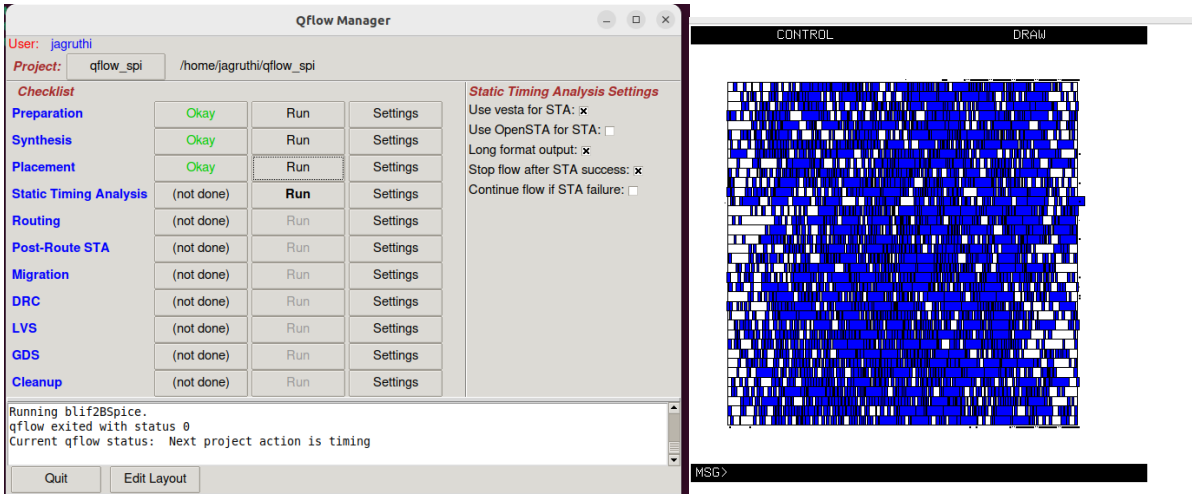
Objectives of Placement

- Assign physical coordinates to each standard cell.
- Minimize total wirelength to reduce signal delay and congestion.
- Avoid cell overlaps to ensure successful routing.
- Prepare a layout that conforms to the chip's area and aspect ratio constraints.

Inputs to Placement

The placement tool in Qflow requires the following inputs:

- Gate-Level Netlist (.blif or .v) generated during synthesis.
- LEF File: Contains abstract layout details of standard cells (cell height, pin locations, obstruction).
- Cell library: osu035 or osu018 with standard cell definitions.
- Floorplan constraints: Defined implicitly or in configuration files (e.g., chip size, row height).




```
6 routing layers
metal1: 367 vertical tracks from -3um with 1um pitch
metal2: 469 vertical tracks from -3.2um with 0.8um pitch
metal3: 367 vertical tracks from -3um with 1um pitch
metal4: 469 vertical tracks from -3.2um with 0.8um pitch
metal5: 367 vertical tracks from -3um with 1um pitch
metal6: 235 vertical tracks from -3.2um with 1.6um pitch
Summary: Total components = 3103
  Fill cells  = 0
  Other cells = 3103
Done with place2def.tcl
```

Static Timing Analysis (STA)

Static Timing Analysis (STA) is a method used to validate the timing performance of a digital circuit without applying test vectors. It ensures that all data paths in the circuit meet required timing constraints (such as setup and hold times), guaranteeing that the circuit functions correctly at the desired clock frequency.

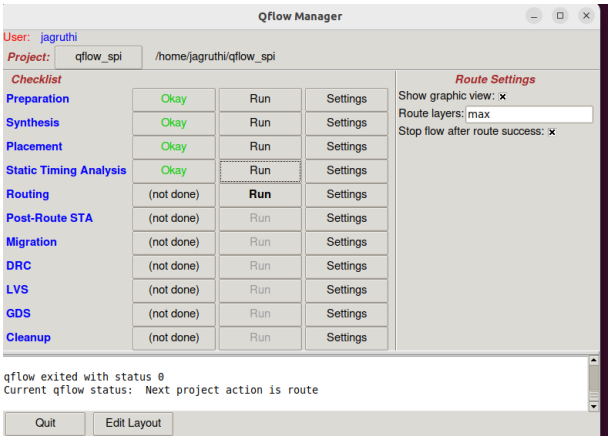
Unlike dynamic simulation, STA does not check for functional correctness but focuses purely on timing correctness under worst-case conditions.

In synchronous designs like SPI, data transfer is controlled by a clock. If a signal arrives too late or too early at a flip-flop:

- It may violate the setup or hold timing.
- This can cause metastability, incorrect data capture, or functional failure.

STA identifies such issues and ensures timing closure—i.e., all paths meet timing constraints.

1. **Setup Time Check**: Ensures data arrives before the clock edge (data must be stable for a duration before the clock).
2. **Hold Time Check**: Ensures data remains stable after the clock edge.
3. **Clock Skew Analysis**: Checks delay difference between clock arrivals at different flip-flops.
4. **Slack Calculation**: Slack = (Required Time) - (Arrival Time)
    - **Positive Slack** → Path meets timing.
    - **Negative Slack** → Timing violation.
5. Setup Time Check: Ensures data arrives before the clock edge (data must be stable for a duration before the clock).
6. Hold Time Check: Ensures data remains stable after the clock edge.
7. Clock Skew Analysis: Checks delay difference between clock arrivals at different flip-flops.
8. Slack Calculation: Slack = (Required Time) - (Arrival Time)
    - Positive Slack → Path meets timing.
    - Negative Slack → Timing violation.




```
Qflow static timing analysis logfile appended on Thursday 26 June 2025 10:18:03 PM IST
Running vesta static timing analysis
vesta --long spi_top.rtlnopwr.v /usr/share/qflow/tech/osu018/osu018_stdcells.lib
-----------------------------------------------
Vesta static timing analysis tool
for qflow 1.3.17
(c) 2013-2018 Tim Edwards, Open Circuit Design
-----------------------------------------------

Parsing library "osu018_stdcells"
End of library at line 6141
Parsing module "spi_top"
Lib read /usr/share/qflow/tech/osu018/osu018_stdcells.lib:  Processed 6142 lines.
Verilog netlist read:  Processed 3127 lines.
Number of paths analyzed:  321

Top 20 maximum delay paths:
Path DFFSR_42/CLK to DFFSR_164/D delay 4156.68 ps
      0.0 ps  wb_clk_i_bF_buf3_bF_buf1:  CLKBUF1_27/Y ->    DFFSR_42/CLK
    642.6 ps                rx_negedge:    DFFSR_42/Q -> NAND3X1_253/C
   1016.1 ps                    _1958_: NAND3X1_253/Y ->    NOR3X1_5/C
   1213.3 ps                    _1985_:    NOR3X1_5/Y -> NAND2X1_118/B
   1320.0 ps                    _1994_: NAND2X1_118/Y -> NAND3X1_263/B
   1549.9 ps                    _1995_: NAND3X1_263/Y -> NAND2X1_120/A
   1753.7 ps                    _1999_: NAND2X1_120/Y ->    NOR3X1_7/C
   1851.4 ps                    _2000_:    NOR3X1_7/Y ->  AOI21X1_66/B
   1959.3 ps                    _2003_:  AOI21X1_66/Y -> OAI21X1_255/A
   2053.5 ps                    _2077_: OAI21X1_255/Y -> OAI21X1_256/A
   2307.3 ps                    _2078_: OAI21X1_256/Y ->   BUFX4_174/A
   2504.8 ps            _2078__bF_buf5:   BUFX4_174/Y -> OAI21X1_271/A
   2600.9 ps                    _2118_: OAI21X1_271/Y ->  OAI22X1_56/A
   2705.8 ps                    _2121_:  OAI22X1_56/Y -> OAI21X1_273/B
   2793.3 ps                    _2122_: OAI21X1_273/Y -> NAND2X1_138/A
   2875.2 ps                    _2136_: NAND2X1_138/Y -> OAI21X1_279/B
   2961.0 ps                    _2138_: OAI21X1_279/Y -> NAND3X1_274/B
   3054.1 ps                     _587_: NAND3X1_274/Y -> OAI21X1_345/A
   3235.9 ps                     _685_: OAI21X1_345/Y ->   BUFX4_274/A
   3444.9 ps             _685__bF_buf3:   BUFX4_274/Y ->    INVX8_14/A
   3519.3 ps                     _686_:    INVX8_14/Y ->   BUFX4_153/A
   3692.4 ps             _686__bF_buf7:   BUFX4_153/Y -> OAI21X1_545/A
   3794.4 ps                    _1160_: OAI21X1_545/Y ->  AOI22X1_73/C
   3872.7 ps                _544__48_:  AOI22X1_73/Y ->   DFFSR_164/D

   clock skew at destination = 184.514
   setup at destination = 99.4674
```

Routing

Routing is the physical design step where the electrical connections (wires) between the standard cells—based on the logical netlist—are created using predefined metal layers of the CMOS process. After placement determines the location of each gate, routing adds the metal tracks and vias to make sure the right signals flow between the gates as specified.

In Qflow, routing is handled by Qrouter, an open-source detail router designed for use with standard-cell layouts.

Objective of Routing

- Connect all nets (wires) in the netlist physically on silicon.
- Minimize delay, congestion, and DRC violations.
- Use metal tracks efficiently across multiple layers.
- Ensure signal integrity (minimize crosstalk and noise).

Routing is performed in two major phases:

1. Global Routing – Plans approximate wire paths across the layout.
2. Detailed Routing – Assigns exact metal tracks and creates connections using vias.

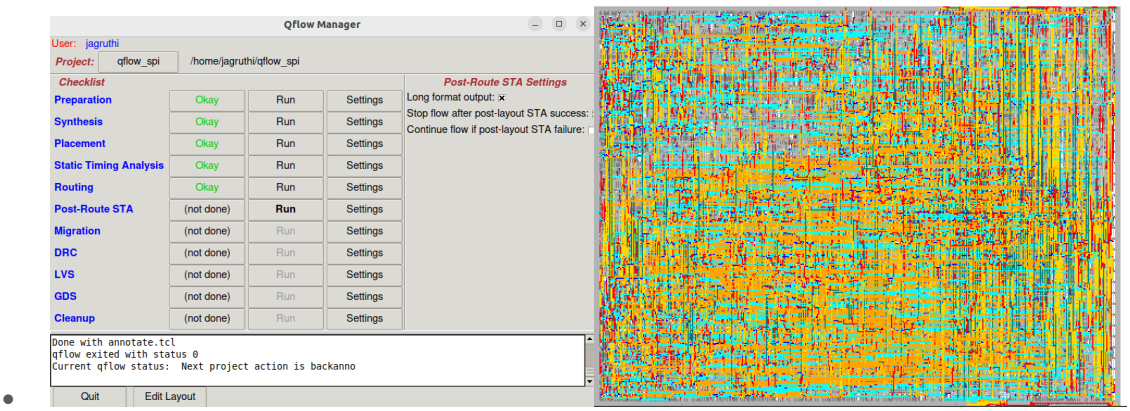
8.2 Routing in Qflow

qflow route spi

This invokes Qrouter, which reads the .def file generated during placement and the standard-cell technology LEF file (osu035/018).

Qrouter uses:

- Preferred routing directions: e.g., horizontal on Metal1, vertical on Metal2.
- Grid-based routing: aligns wires to routing tracks.
- Design rules: from the technology file to avoid shorts, spacing violations, etc.
- Technology rules: Defines via sizes, metal widths, spacing, etc.




```
----------------------------------------------
Progress: Stage 3 total routes completed: 22448
No failed routes!
----------------------------------------------
*** Writing DEF file spi_top_route.def
emit_routes():  DEF file has 3152 nets and 2 specialnets.
but qrouter wants to write 3152 nets and specialnets.
----------------------------------------------
Final: No failed routes!
----------------------------------------------
```

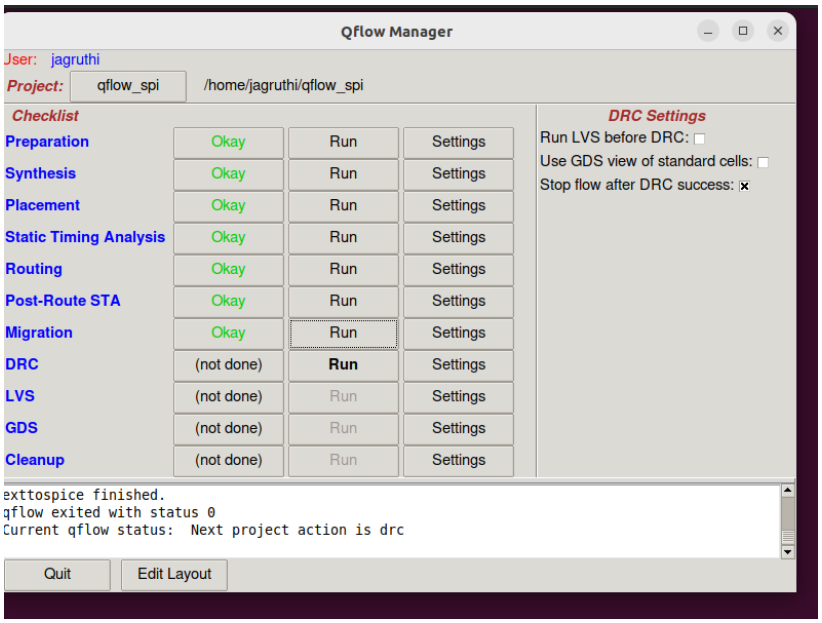
```
Top 20 maximum delay paths:
Path DFFSR_42/CLK to DFFSR_180/D delay 4398.21 ps
      1.3 ps  wb_clk_i_bF_buf3_bF_buf1:  CLKBUF1_27/Y ->    DFFSR_42/CLK
    685.5 ps                rx_negedge:    DFFSR_42/Q -> NAND3X1_253/C
   1067.1 ps                    _1958_: NAND3X1_253/Y ->    NOR3X1_5/C
   1276.9 ps                    _1985_:    NOR3X1_5/Y -> NAND2X1_118/B
   1386.9 ps                    _1994_: NAND2X1_118/Y -> NAND3X1_263/B
   1627.7 ps                    _1995_: NAND3X1_263/Y -> NAND2X1_120/A
   1840.0 ps                    _1999_: NAND2X1_120/Y ->    NOR3X1_7/C
   1941.5 ps                    _2000_:    NOR3X1_7/Y ->  AOI21X1_66/B
   2055.8 ps                    _2003_:  AOI21X1_66/Y -> OAI21X1_255/A
   2153.0 ps                    _2077_: OAI21X1_255/Y -> OAI21X1_256/A
   2453.5 ps                    _2078_: OAI21X1_256/Y ->   BUFX4_174/A
   2654.5 ps            _2078__bF_buf5:   BUFX4_174/Y -> OAI21X1_271/A
   2754.8 ps                    _2118_: OAI21X1_271/Y ->  OAI22X1_56/A
   2863.3 ps                    _2121_:  OAI22X1_56/Y -> OAI21X1_273/B
   2956.1 ps                    _2122_: OAI21X1_273/Y -> NAND2X1_138/A
   3040.8 ps                    _2136_: NAND2X1_138/Y -> OAI21X1_279/B
   3225.1 ps                    _2138_: OAI21X1_279/Y -> NAND3X1_274/B
   3431.4 ps                     _587_: NAND3X1_274/Y -> OAI21X1_345/A
   3654.3 ps                     _685_: OAI21X1_345/Y ->   BUFX4_274/A
   3733.9 ps            _685__bF_buf3:   BUFX4_274/Y ->    INVX8_14/A
   3906.8 ps                     _686_:    INVX8_14/Y ->   BUFX4_159/A
   4007.4 ps            _686__bF_buf1:   BUFX4_159/Y -> OAI21X1_504/A
   4094.9 ps                    _1062_: OAI21X1_504/Y ->  AOI22X1_69/C
                          _544__64_:  AOI22X1_69/Y ->   DFFSR_180/D

   clock skew at destination = 199.755
   setup at destination = 103.564
```

Terminal

MIGRATION

RC EXTRACTION DETAILS OF THE SPI PROJECT CAN BE OBSERVED IN THE MIGRATION LOG FILE.




```
Generating LEF output spi_top.lef for cell spi_top:
Diagnostic:  Write LEF header for cell spi_top
Diagnostic:  Writing LEF output for cell spi_top
Diagnostic:  Scale value is 0.100000
Extracting XNOR2X1 into XNOR2X1.ext:
Extracting XOR2X1 into XOR2X1.ext:
Extracting NOR3X1 into NOR3X1.ext:
Extracting OR2X2 into OR2X2.ext:
Extracting INVX4 into INVX4.ext:
Extracting NOR2X1 into NOR2X1.ext:
Extracting AND2X2 into AND2X2.ext:
Extracting BUFX2 into BUFX2.ext:
Extracting MUX2X1 into MUX2X1.ext:
Extracting INVX1 into INVX1.ext:
Extracting OAI22X1 into OAI22X1.ext:
Extracting NAND2X1 into NAND2X1.ext:
Extracting OAI21X1 into OAI21X1.ext:
Extracting AOI22X1 into AOI22X1.ext:
Extracting DFFSR into DFFSR.ext:
Extracting INVX2 into INVX2.ext:
Extracting CLKBUF1 into CLKBUF1.ext:
Extracting FILL into FILL.ext:
Extracting NAND3X1 into NAND3X1.ext:
Extracting AOI21X1 into AOI21X1.ext:
Extracting INVX8 into INVX8.ext:
Extracting BUFX4 into BUFX4.ext:
Extracting spi_top into spi_top.ext:
```

Design Rule Check (DRC)

Design Rule Check (DRC) is a crucial verification step in the physical design flow. It ensures that the final layout of the integrated circuit (IC) complies with the geometric and spacing constraints imposed by the semiconductor fabrication process. These constraints are defined by the foundry (e.g., minimum metal width, spacing between wires, via enclosures, etc.) and must be strictly followed for a successful chip fabrication.

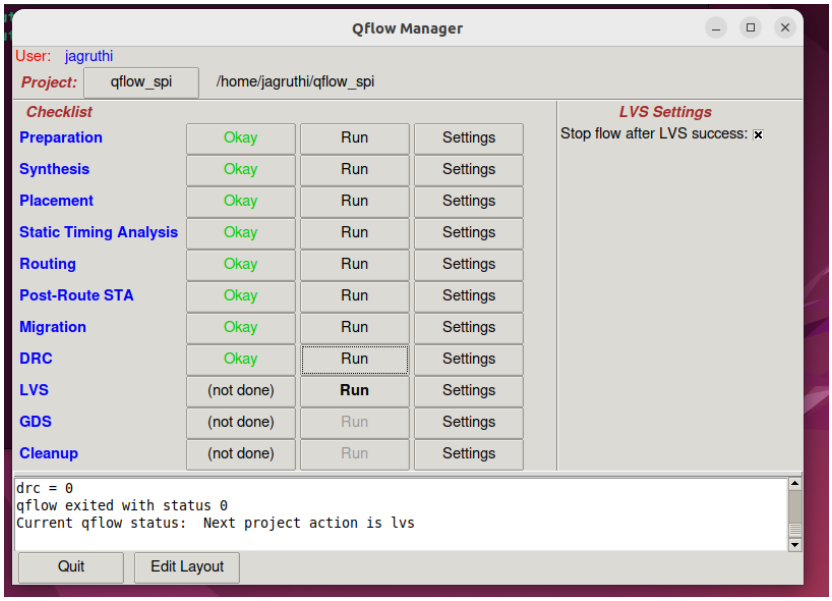
In this project, DRC is performed using Magic VLSI, an open-source layout editor and rule-checking tool integrated into the Qflow toolchain.

Objective of DRC

The primary goals of DRC are:

- To verify that all layout geometries (metal layers, contacts, vias, active regions) follow the design rules of the selected process (e.g., osu035 or osu018).
- To ensure fabrication readiness by preventing violations that can cause yield issues, short circuits, or open circuits.
- To catch layout-related errors before tape-out, saving time and cost.

```
spi_top: 10000 rects
spi_top: 20000 rects
spi_top: 30000 rects
spi_top: 40000 rects
spi_top: 50000 rects
Processing timestamp mismatches: XOR2X1, XNOR2X1, OR2X2, OAI22X1, OAI21X1, NOR3X1, NOR2X1, NAND3X1
NAND2X1, MUX2X1, INVX8, INVX4, INVX2, INVX1, FILL, DFFSR, CLKBUF1, BUFX4, BUFX2, AOI22X1, AOI21X1,
AND2X2.
drc = 0
```



LVS

LAYOUT VERSUS SCHEMATIC (LVS) IS A CRITICAL VERIFICATION STEP IN THE VLSI BACKEND DESIGN PROCESS. IT CHECKS WHETHER THE PHYSICAL LAYOUT OF THE DESIGN (AFTER PLACEMENT AND ROUTING) ACCURATELY IMPLEMENTS THE ORIGINAL LOGICAL NETLIST (FROM SYNTHESIS). IN OTHER WORDS, IT VERIFIES THAT "WHAT YOU DRAW" IS EXACTLY "WHAT YOU INTENDED LOGICALLY."

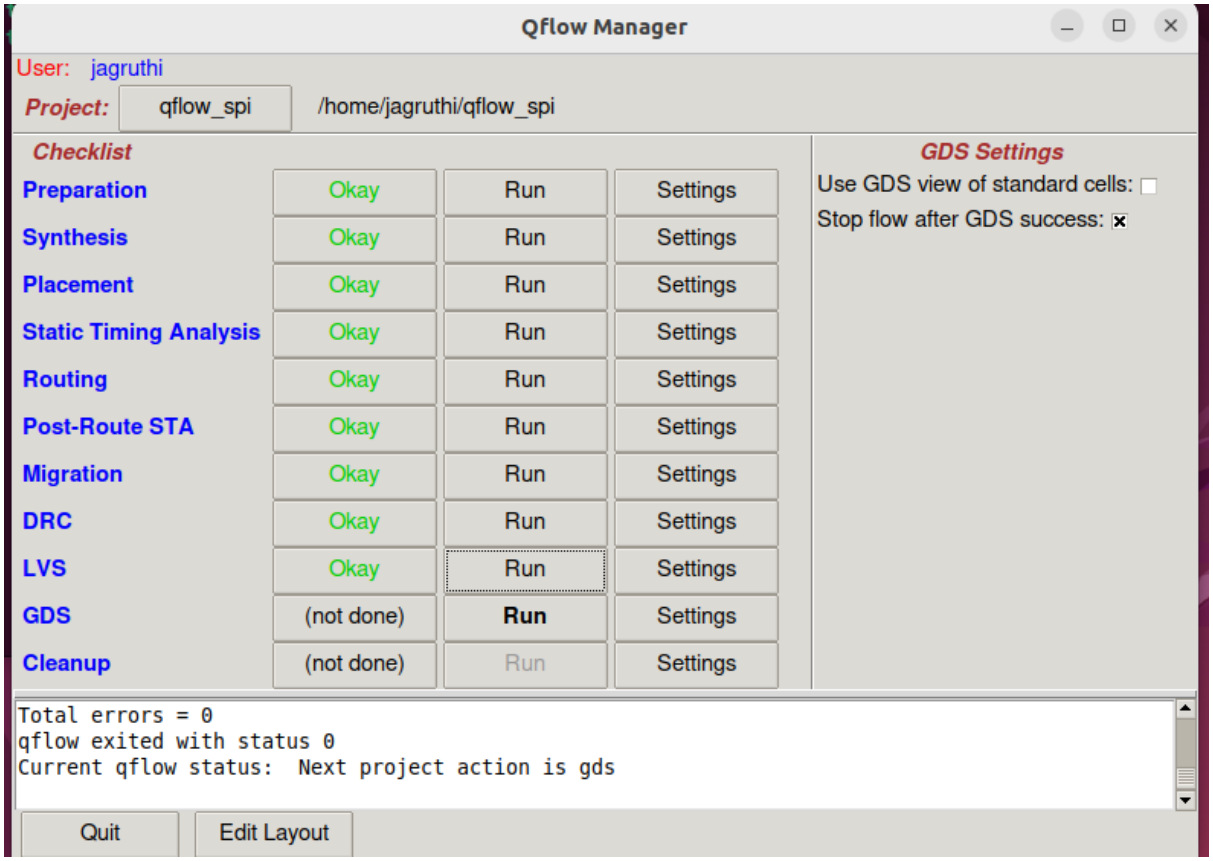
LVS ENSURES FUNCTIONAL CORRECTNESS, WHEREAS DRC ENSURES MANUFACTURABILITY.

PURPOSE OF LVS

THE GOAL OF LVS IS TO:

- COMPARE THE SCHEMATIC NETLIST (FROM SYNTHESIS/YOSYS) WITH THE EXTRACTED NETLIST (FROM THE ROUTED LAYOUT).
- ENSURE:
    - ALL DEVICES MATCH (FLIP-FLOPS, GATES, ETC.)
    - ALL CONNECTIONS (NETS) MATCH
    - NO EXTRA OR MISSING COMPONENTS EXIST
- CONFIRM THAT THE LAYOUT WILL FUNCTION EXACTLY AS THE RTL/SCHEMATIC INTENDS.

```
Subcircuit pins:
Circuit 1: XOR2X1                          |Circuit 2: XOR2X1
-------------------------------------------|-------------------
A                                          |A
B                                          |B
gnd                                        |gnd
Y                                          |Y
vdd                                        |vdd
-----------------------------------------------------------------
Cell pin lists are equivalent.
Device classes XOR2X1 and XOR2X1 are equivalent.
Subcircuit summary:
Circuit 1: spi_top                         |Circuit 2: spi_top
-------------------------------------------|-------------------------------------------
AND2X2 (29)                                |AND2X2 (29)
OAI21X1 (711)                              |OAI21X1 (711)
MUX2X1 (48)                                |MUX2X1 (48)
DFFSR (253)                                |DFFSR (253)
NAND2X1 (262)                              |NAND2X1 (262)
AOI22X1 (90)                               |AOI22X1 (90)
AOI21X1 (224)                              |AOI21X1 (224)
NAND3X1 (276)                              |NAND3X1 (276)
OAI22X1 (123)                              |OAI22X1 (123)
INVX2 (160)                                |INVX2 (160)
BUFX4 (277)                                |BUFX4 (277)
NOR2X1 (263)                               |NOR2X1 (263)
OR2X2 (13)                                 |OR2X2 (13)
XNOR2X1 (14)                               |XNOR2X1 (14)
INVX8 (26)                                 |INVX8 (26)
INVX1 (158)                                |INVX1 (158)
NOR3X1 (9)                                 |NOR3X1 (9)
CLKBUF1 (67)                               |CLKBUF1 (67)
XOR2X1 (6)                                 |XOR2X1 (6)
BUFX2 (82)                                 |BUFX2 (82)
INVX4 (12)                                 |INVX4 (12)
Number of devices: 3103                    |Number of devices: 3103
Number of nets: 3150                       |Number of nets: 3150
-----------------------------------------------------------------------------------------
Netlists match uniquely.
```


Qflow Manager
User: jagruthi
Project: qflow_spi
/home/jagruthi/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 Okay Run Settings
Migration Okay Run Settings
DRC Okay Run Settings
LVS Okay Run Settings
GDS (not done) Run Settings
Cleanup (not done) Run Settings
GDS Settings
Use GDS view of standard cells:
Stop flow after GDS success:
Total errors = 0
qflow exited with status 0
Current qflow status: Next project action is gds
Quit
Edit Layout

GDS

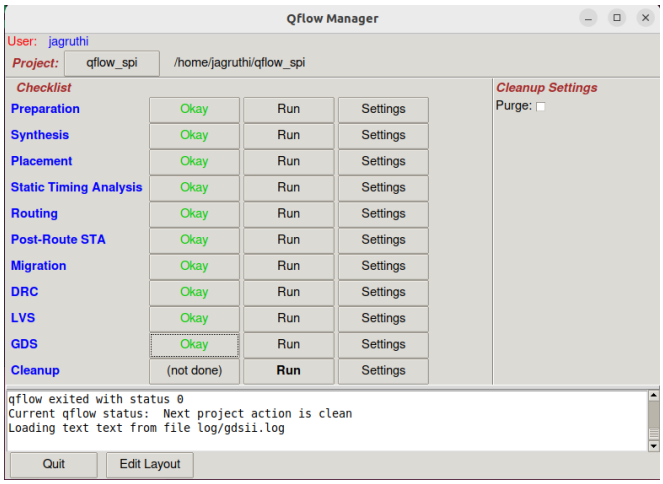
GDSII (Graphic Data System II), also called Stream format, is the industry-standard file format used to represent the final layout of an integrated circuit (IC). It contains geometric data describing all layers (metal, polysilicon, diffusion, etc.) in a binary format that fabrication tools can read.

Once DRC and LVS checks are clean, GDSII is generated as the final output of the layout process—this file is what is sent to the foundry for silicon manufacturing.

Purpose of GDSII

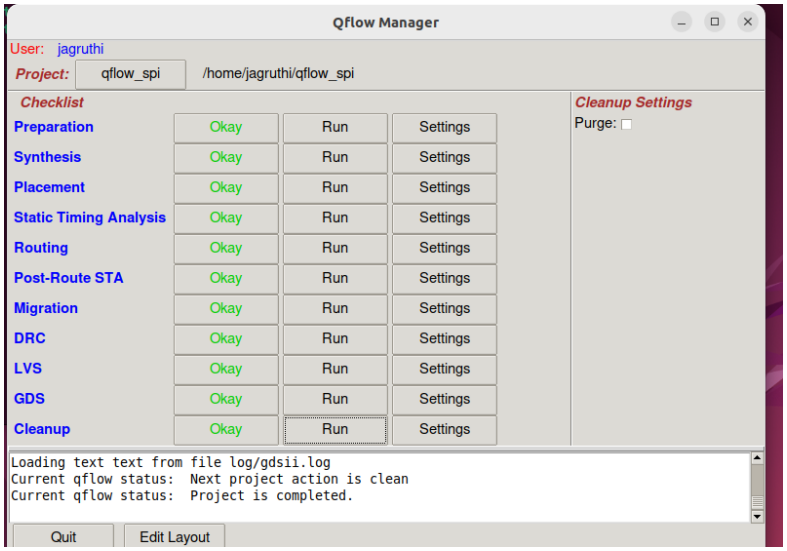
- Encodes all physical layers and structures (wires, vias, transistors, etc.) in a compact binary format.
- Used by fabrication tools for mask generation and silicon printing.
- Serves as the final deliverable of the physical design cycle.
- Can also be opened in GDS viewers for final visual inspection.

```
Generating output for cell DFFSR
Generating output for cell CLKBUF1
Generating output for cell BUFX2
Generating output for cell FILL
Generating output for cell OAI21X1
Generating output for cell INVX1
Generating output for cell NAND3X1
Generating output for cell NAND2X1
Generating output for cell AOI22X1
Generating output for cell OAI22X1
Generating output for cell INVX8
Generating output for cell INVX2
Generating output for cell BUFX4
Generating output for cell NOR2X1
Generating output for cell AOI21X1
Generating output for cell AND2X2
Generating output for cell MUX2X1
Generating output for cell OR2X2
Generating output for cell INVX4
Generating output for cell NOR3X1
Generating output for cell XOR2X1
Generating output for cell XNOR2X1
Generating output for cell spi_top
```



CLEANUP:

THE CLEANUP STEP IN THE QFLOW-BASED PHYSICAL DESIGN PROCESS INVOLVES REMOVING UNNECESSARY INTERMEDIATE FILES, ORGANIZING FINAL RESULTS, AND ENSURING THAT ONLY THE ESSENTIAL OUTPUTS (LIKE LAYOUT FILES, NETLISTS, REPORTS, AND GDS) REMAIN. THIS HELPS KEEP YOUR PROJECT DIRECTORY CLEAN, SAVES DISK SPACE, AND SIMPLIFIES BACKUP OR SUBMISSION.




```
Top 20 maximum delay paths:
Path DFFSR_42/CLK to DFFSR_240/D delay 4160.96 ps
0.0 ps  wb_clk_i_bF_buf3_bF_buf2:  CLKBUF1_26/Y ->    DFFSR_42/CLK
642.6 ps                rx_negedge:    DFFSR_42/Q -> NAND3X1_253/C
1016.1 ps                    _1958_: NAND3X1_253/Y ->    NOR3X1_5/C
1213.3 ps                    _1985_:    NOR3X1_5/Y -> NAND2X1_118/B
1320.0 ps                    _1994_: NAND2X1_118/Y -> NAND3X1_263/B
1549.9 ps                    _1995_: NAND3X1_263/Y -> NAND2X1_120/A
1753.7 ps                    _1999_: NAND2X1_120/Y ->    NOR3X1_7/C
1851.4 ps                    _2000_:    NOR3X1_7/Y ->  AOI21X1_66/B
1959.3 ps                    _2003_:  AOI21X1_66/Y -> OAI21X1_255/A
2053.5 ps                    _2077_: OAI21X1_255/Y -> OAI21X1_256/A
2307.3 ps                    _2078_: OAI21X1_256/Y ->   BUFX4_177/A
2499.2 ps            _2078__bF_buf2:   BUFX4_177/Y -> OAI21X1_271/A
2594.1 ps                    _2118_: OAI21X1_271/Y ->  OAI22X1_56/A
2698.8 ps                    _2121_:  OAI22X1_56/Y -> OAI21X1_273/B
2786.3 ps                    _2122_: OAI21X1_273/Y -> NAND2X1_138/A
2868.2 ps                    _2136_: NAND2X1_138/Y -> OAI21X1_279/B
2954.0 ps                    _2138_: OAI21X1_279/Y -> NAND3X1_274/B
3047.1 ps                     _587_: NAND3X1_274/Y -> OAI21X1_345/A
3228.9 ps                     _685_: OAI21X1_345/Y ->   BUFX4_273/A
3434.2 ps            _685__bF_buf4:   BUFX4_273/Y ->    INVX8_14/A
3507.3 ps                     _686_:    INVX8_14/Y ->   BUFX4_156/A
3679.8 ps            _686__bF_buf4:   BUFX4_156/Y -> OAI21X1_354/A
3781.8 ps                     _718_: OAI21X1_354/Y ->  AOI22X1_38/C
3860.0 ps                 _544__124_:  AOI22X1_38/Y ->   DFFSR_240/D

clock skew at destination = 206.651
setup at destination = 94.2984
```

OVERALL LAYOUT: