MAVEN SILICON

Centre of Excellence in Semicon

Project On Physical design implementation of SPI Protocol Using Qflow

Name- S Steven Joshua

Course-Maven Silicon PD internship

Batch no-B2B PDI-01

TABLE OF CONTENT

INTRODUCTION

Qflow – An Open-Source Digital Synthesis Flow

A digital synthesis flow is a set of tools and methods used to turn a circuit design written in a high-level behavioral language like Verilog or VHDL into a netlist structure, which can further be configuration data for an FPGA target like a Xilinx or Altera chip, or a format in a chip fabrication process technology, that can be used for a physical, standard-cell CMOS layout.

There are many proprietary flows available from EDA tool vendors, and while they are typically not open source, they are generally distributed for free (presumably on condition of use with the respective vendor's FPGA hardware).

Qflow is a complete flow that supports full synthesis of digital circuits starting from Verilog source and ending in a physical layout in a simple open synthesis process. It supports only a restricted set of Verilog features appropriate to simple application of a synthesis flow. The toolchain includes components from the CellLibrary project, graywolf, and other academic tools and some commercial-quality tools and wrappers. The commercial tools mentioned here require licensing, and have limited access. These tools are all built around an open standard interface for synthesis, and are ideal for use by students and engineering classes.

Qflow consists of the following tools:

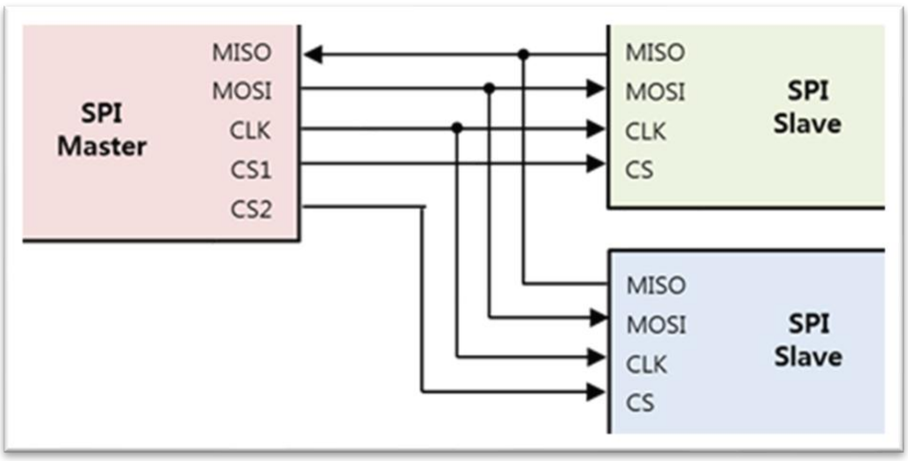
- Yosys:- RTL Sythesis
- Qrouter:- Routing
- Grayflow:- Placement
- Magic:- VLSI Layout Tool
- Netgen:-LVS Layout vs Schematic
- OpenSta:- Static Timing Analysis Tool

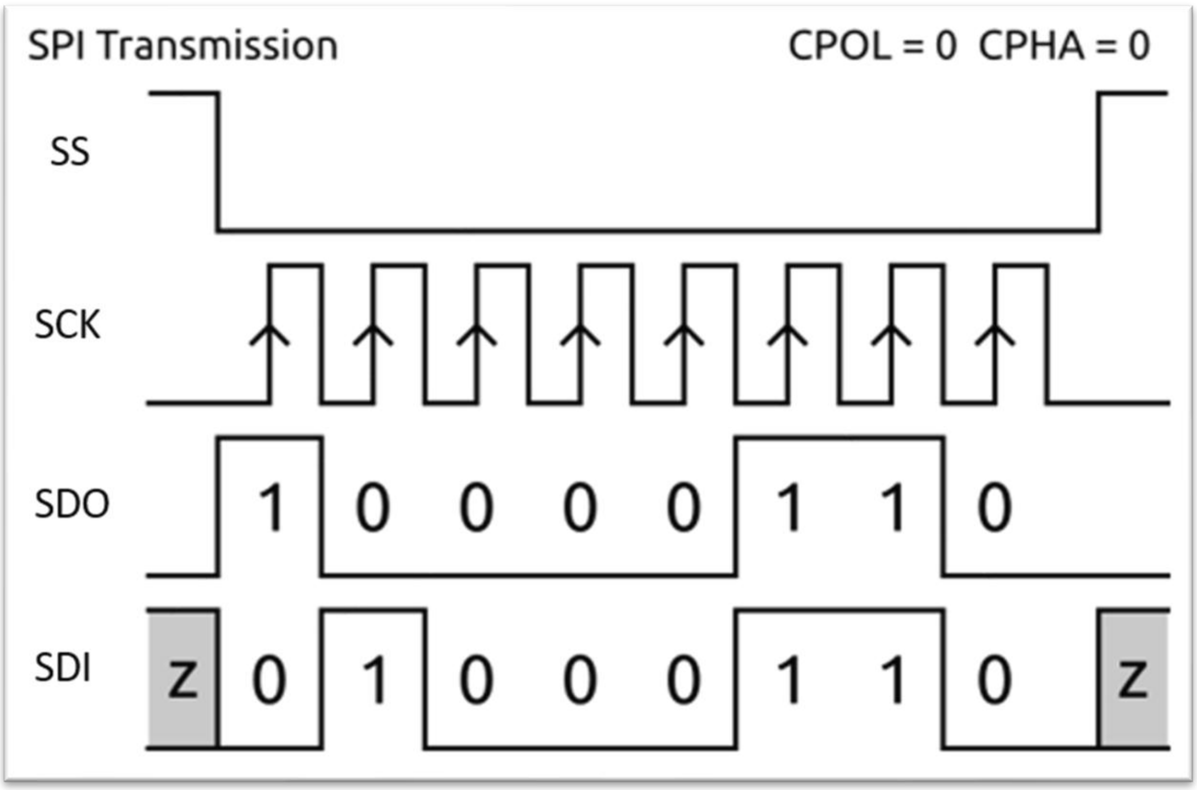
SERIAL PERIPHERAL INTERFACE

Serial Peripheral Interface, or SPI, is a very common communication protocol used for short-distance communication between devices. A standard SPI bus consists of 4 signals: Master Out Slave In (MOSI), Master In Slave Out (MISO), SPI clock (SCLK), and Slave Select (SS). Unlike an asynchronous serial interface, SPI is a synchronous bus. An SPI bus has one master and one or more slaves. The master can talk to any slave on the bus, each slave select signal is unique. Each slave bit on the bus must be unique (either wired separately or through some address selection logic to reduce wiring complexity in larger systems). When the master clock signal is high, both devices detect and capture data on the falling (or) rising edge. Timing is crucial in that the clock is faster than the maximum frequency for all devices involved.

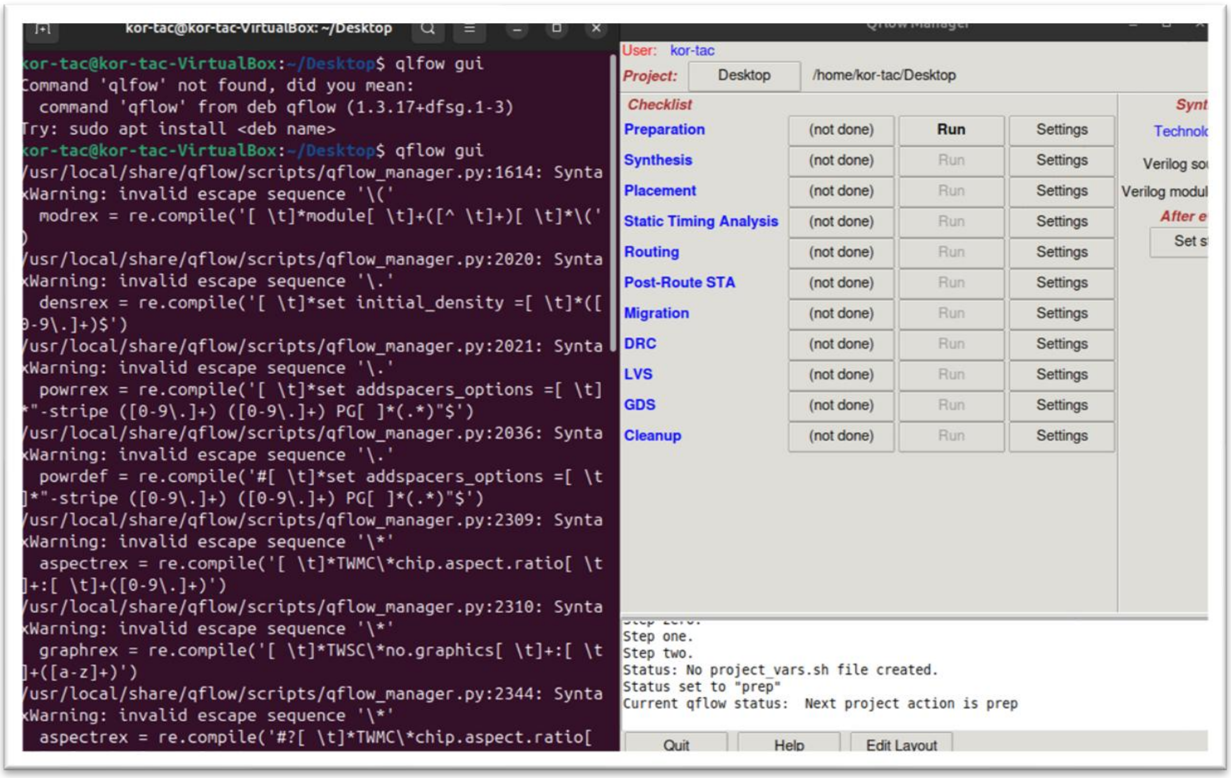
When the master of the SPI bus wants to initiate a transfer, it must first pull the SS signal low for the slave it wants to communicate with. Once the SS signal is low, data will start transferring on the bus. The master is the only one to start sending data.

There are 4 different SPI bus standards that deal with the SCLK signal. The modes are defined using two parameters, CPOL and CPHA. CPOL stands for Clock Polarity and determines the default (idle) level of the SCLK signal when the bus is idle. CPHA stands for Clock Phase and determines which edge of the clock is used to sample incoming data. The slave must be programmed with the correct parameters to avoid capture accordingly. The most common settings are CPOL=0 (idle low) and CPHA=0 (sample rising edge).


SPI Master
MISO
MOSI
CLK
CS1
CS2
MISO
MOSI
CLK
CS
SPI Slave
MISO
MOSI
CLK
CS
SPI Slave


SPI Transmission
CPOL = 0 CPHA = 0
SS
SCK
SDO
1 0 0 0 0 1 1 0
SDI
Z 0 1 0 0 0 1 1 0 Z

INVOKE THE TOOL



```
kor-tac@kor-tac-VirtualBox:~/Desktop$ qlfow gui
Command 'qlfow' not found, did you mean:
  command 'qflow' from deb qflow (1.3.17+dfsg.1-3)
Try: sudo apt install <deb name>
kor-tac@kor-tac-VirtualBox:~/Desktop$ qflow gui
/usr/local/share/qflow/scripts/qflow_manager.py:1614: Synta
xWarning: invalid escape sequence '\('
  modrex = re.compile('[ \t]*module[ \t]+([^ \t]+)[ \t]*\('
)
/usr/local/share/qflow/scripts/qflow_manager.py:2020: Synta
xWarning: invalid escape sequence '\.'
  densrex = re.compile('[ \t]*set initial_density =[ \t]*([
0-9\.]+)$')
/usr/local/share/qflow/scripts/qflow_manager.py:2021: Synta
xWarning: invalid escape sequence '\.'
  powrrex = re.compile('[ \t]*set addspacers_options =[ \t]
*"-stripe ([0-9\.]+) ([0-9\.]+) PG[ ]*(.*)"$')
/usr/local/share/qflow/scripts/qflow_manager.py:2036: Synta
xWarning: invalid escape sequence '\.'
  powrdef = re.compile('#[ \t]*set addspacers_options =[ \t
]*"-stripe ([0-9\.]+) ([0-9\.]+) PG[ ]*(.*)"$')
/usr/local/share/qflow/scripts/qflow_manager.py:2309: Synta
xWarning: invalid escape sequence '\*'
  aspectrex = re.compile('[ \t]*TWMC\*chip.aspect.ratio[ \t
]+:[ \t]+([0-9\.]+)')
/usr/local/share/qflow/scripts/qflow_manager.py:2310: Synta
xWarning: invalid escape sequence '\*'
  graphrex = re.compile('[ \t]*TWSC\*no.graphics[ \t]+:[ \t
]+([a-z]+)')
/usr/local/share/qflow/scripts/qflow_manager.py:2344: Synta
xWarning: invalid escape sequence '\*'
  aspectrex = re.compile('#?[ \t]*TWMC\*chip.aspect.ratio[
```

User: kor-tac

Project: Desktop /home/kor-tac/Desktop

Checklist			
Preparation	(not done)	Run	Settings
Synthesis	(not done)	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

```
Step one.
Step two.
Status: No project_vars.sh file created.
Status set to "prep"
Current qflow status:  Next project action is prep
```

Qflow GUI is a graphical user interface designed to work with the QFlow RTL (Register Design Terminal) toolchain. It provides a user-friendly interface for interacting with Qflow's tools and processes, giving a simple step-by-step view of designing digital circuits on Linux systems. The GUI is intended to make the powerful features of the Qflow toolchain more accessible to users who prefer visual interfaces over command-line operations.

Qflow GUI tool will have each and every sub tool. We have to run each tool after succeeding the previous tools. By clicking the run option we can execute individual tool used in the flow. Execution without an error message is seen as OKAY or will result in FAIL.

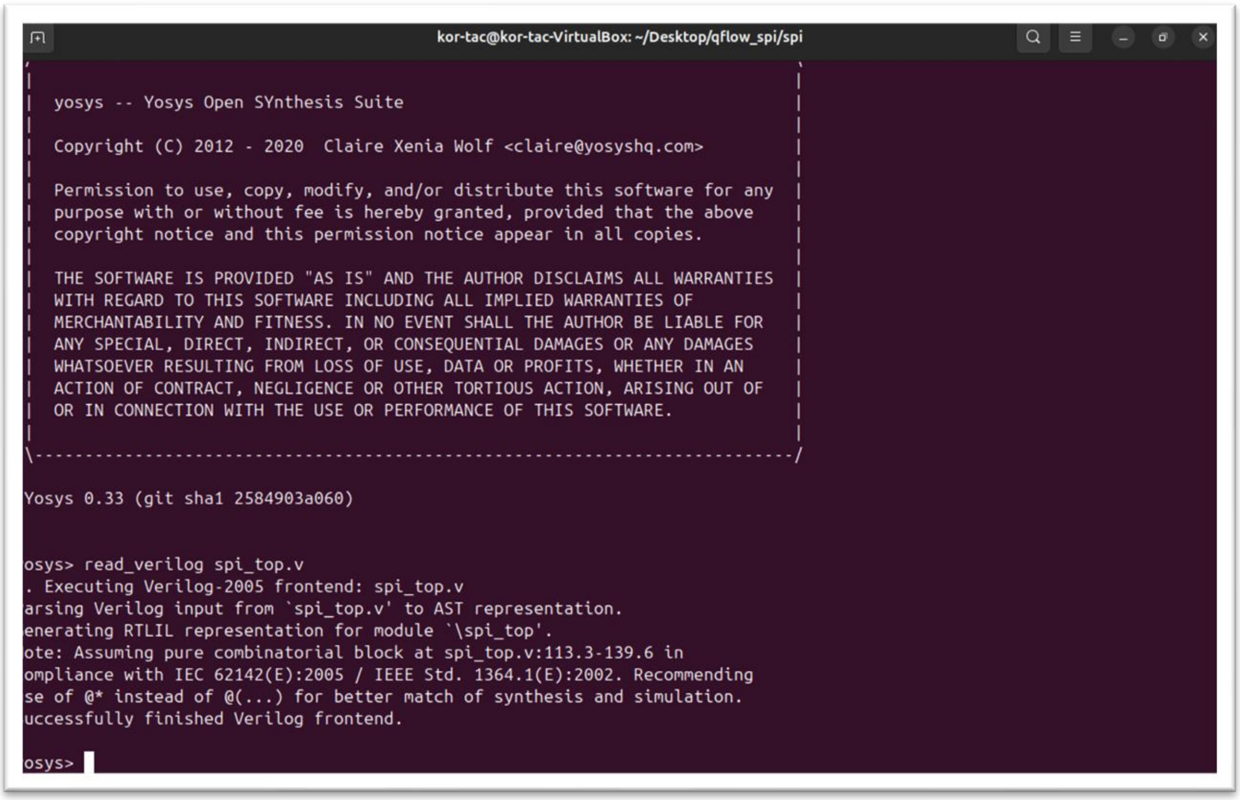
If we follow below steps to invoke the other and to run the tool, by doing this we check the project and design:-

First before invoking we have to select the Technology File (osu018).

After selecting we have to select the top module of Verilog Source File i.e., its extension. (As SPI project we have to select "spi_top"). After this process you need to invoke the Verilog module name i.e. "spi_top" in same manner without an extension.

Note: After this Preparation tab → Check the project directory we can see 4 new folders have been created automatically. The folders are: layout, source, log, synthesis.

SYNTHESIS



```
|                                                                            |
|  yosys -- Yosys Open SYnthesis Suite                                       |
|                                                                            |
|  Copyright (C) 2012 - 2020  Claire Xenia Wolf <claire@yosyshq.com>         |
|                                                                            |
|  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.            |
|                                                                            |
\----------------------------------------------------------------------------/

Yosys 0.33 (git sha1 2584903a060)


osys> read_verilog spi_top.v
. Executing Verilog-2005 frontend: spi_top.v
arsing Verilog input from `spi_top.v' to AST representation.
enerating RTLIL representation for module `\spi_top'.
ote: Assuming pure combinatorial block at spi_top.v:113.3-139.6 in
ompliance with IEC 62142(E):2005 / IEEE Std. 1364.1(E):2002. Recommending
se of @* instead of @(...) for better match of synthesis and simulation.
uccessfully finished Verilog frontend.

osys>
```

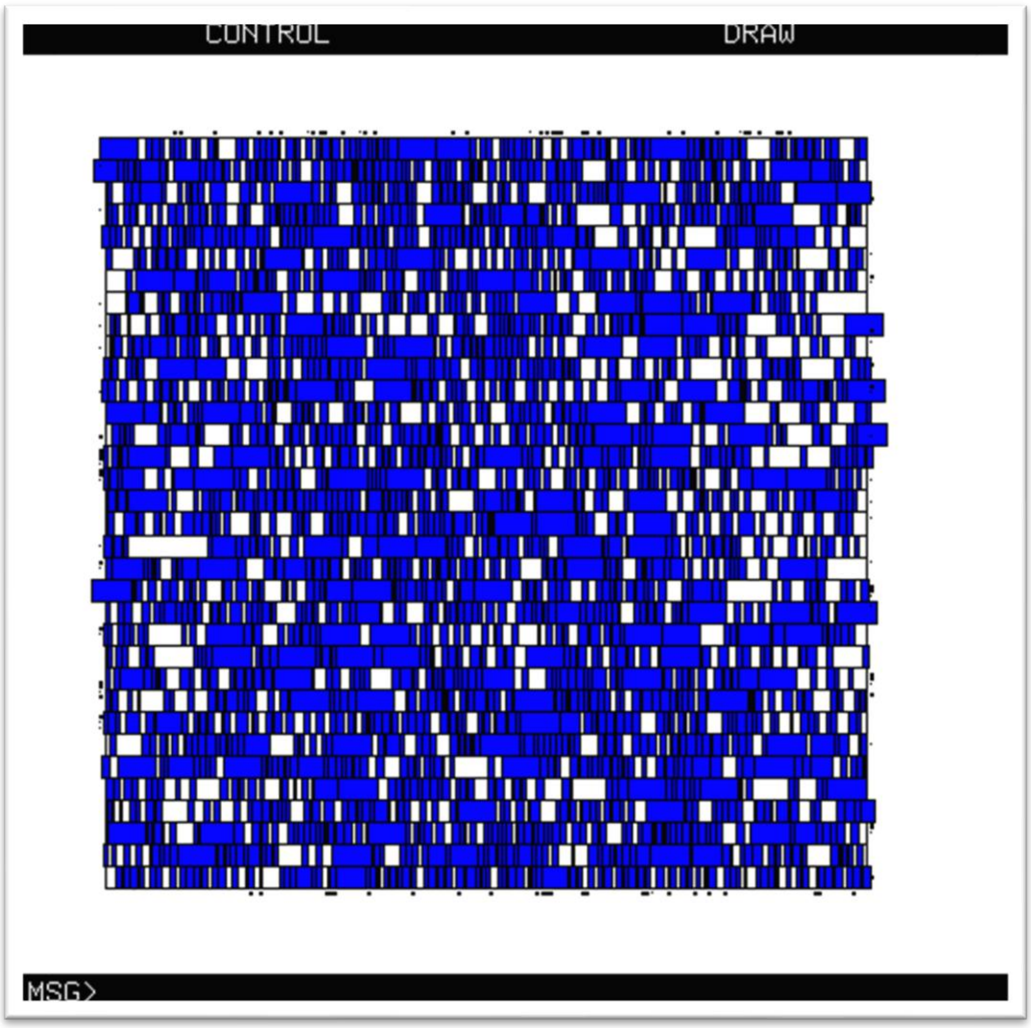
Synthesis is a critical step in the digital design process, where high-level hardware description language (HDL), such as Verilog or VHDL, are converted into a gate-level representation to interact with physical hardware, such as FPGAs or ASICs. In Qflow, synthesis is performed using the Yosys tool, and Qflow GUI provides a graphical interface to manage and execute the synthesis process.

The most important thing we have to note here is: to mention the project path to avoid the error just before, so we must set the working file path. It is the first place to verify created source folder, output, etc. Then Run the Synthesis Tool. We have to wait till we can get Output.

YOSYS

Yosys is an essential open-source synthesis tool within the Qflow FPGA design toolchain. It is responsible for converting high-level hardware design languages (HDL) code, such as Verilog or VHDL, into a gate-level design suitable for placement & logical layout. The synthesis component of Qflow, Yosys performs optimization and technology mapping, generating RTL designs in a format suitable for further processing in the FPGA design flow. It generates intermediate files like Berkeley Logic Interchange Format (BLIF), which are necessary for the subsequent logic placement, routing, and physical synthesis stages. Yosys is known for its lightweight core structure and active development, making it a powerful and cost-effective tool for FPGA implementation. The community heavily maintains Yosys with a specific focus and set of plug-in tools like the ABC tool, that align well with Yosys and the desired output format. Proper installation and configuration of Yosys are crucial, and it is supported by extensive documentation and active forums, which provide valuable resources for troubleshooting and optimization.

PLACEMENT



Placement is the process of placing all the standard cells that are present in the netlist in the real-time core area. Tool also optimizes the design while placing. During placement trial routing also can be done by the tool.

Placement is done by the tool in 3 different techniques, which are:

1. Timing driven placement – Placement happens with timing as a priority
2. Congestion driven placement – Placement happens with routing as a priority
3. Power driven placement – Placement happens with power as a priority

Goals of placement:

- **QoR goals:**
  - Timing, area and power optimization
  - Minimize congestion and optimization hotspots
- **Minimize cell density, pin density**
- **No DRC (Design Rule Check)**
  - No being DRC's
- **Placement objectives:**
  - Physical Half Periphery (Hp)
  - Placement Db File

Placement Steps:

- Pre-Placement
- Placement
- Post-Placement

GRAYWOLF

In the Qflow FPGA design toolchain, Graywolf is the placement tool responsible for placing the logic cells onto the FPGA fabric or ASIC layout. Graywolf employs a force-directed algorithm for optimizing the spatial arrangement of the cells to improve layout proximity and minimize signal delays. Its primary objective is to minimize signal delay and congestion by strategically positioning cells. It works by determining the best possible location for each cell based on wire length and signal interactions. It also balances the layout, considering factors such as cell density and interconnect distances, to ensure the design can route fully with minimal congestion. Graywolf also supports region constraints and handles design hierarchy. By providing an effective placement solution, Graywolf helps set the stage for efficient routing.

STATIC TIMING ANALYSIS

Static Timing Analysis (STA) is a crucial technique used in digital design to ensure circuits perform as expected, especially under varying conditions. Unlike simulations, STA doesn't need input vectors. Instead, it analyzes the timing paths in a design by checking all possible routes that signals can take and evaluates delays.

It verifies setup and hold times, clock skews, and propagation delays. This helps avoid timing-related errors that could lead to circuit failure. STA is especially important for high-speed digital circuits, as it helps identify critical timing paths that may cause delays or errors.

STA provides a way to validate whether a design meets its required timing constraints without needing to simulate every possible input combination. It ensures that performance and reliability requirements are satisfied.

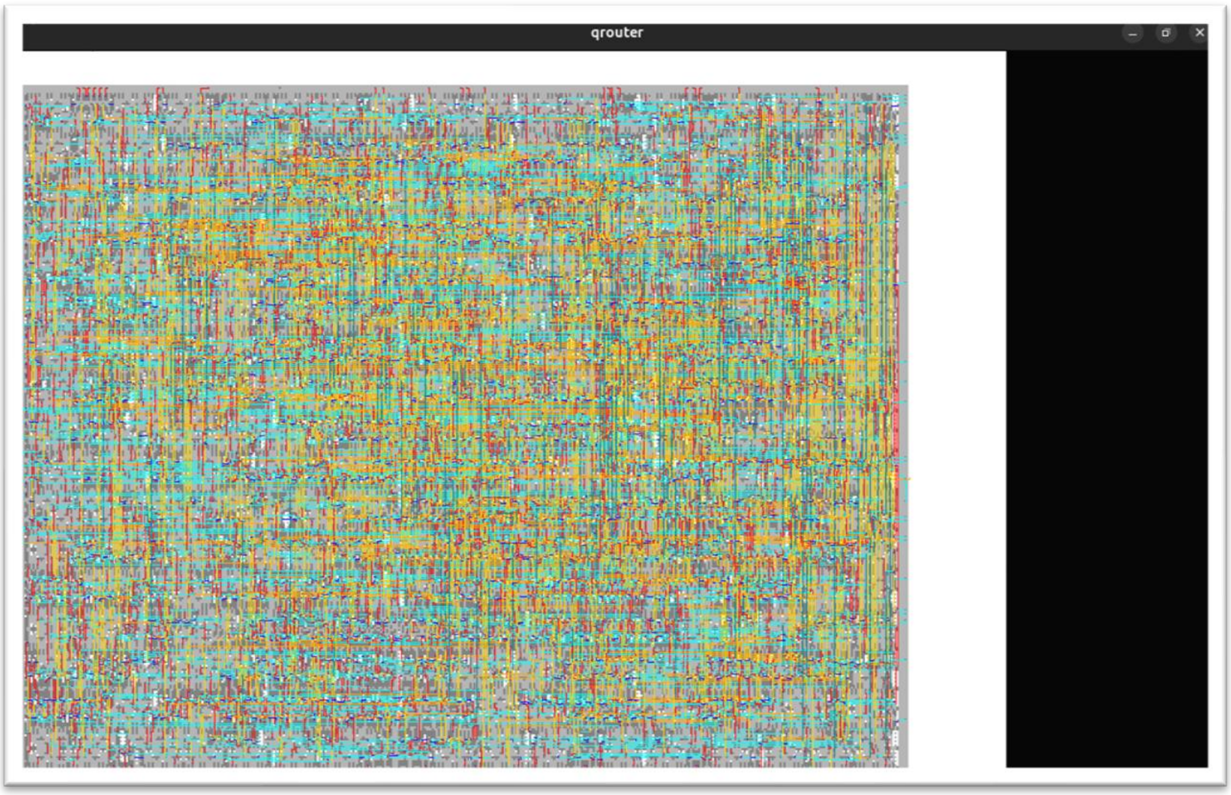
OpenSTA

OpenSTA is a free, open-source tool for performing STA. It's integrated into the Qflow toolchain and provides a way to do in-depth timing checks. It features a user-friendly interface, supports both GUI and command-line usage, and includes essential features such as:

- Path-based analysis
- Timing constraint checking
- Delay estimation

The goal is to ensure timing constraints are satisfied, optimizing the design for performance and reliability.

ROUTING



In the Qflow FPGA design flow, routing is a key phase. It’s the process where physical wires (or interconnections) are laid out to connect logic blocks — think of it as building roads between cities.

The router takes the placement information and works out the actual wiring paths needed to join various signals, making sure they don’t interfere with each other or cause problems like delays or signal conflicts.

Routing needs to:

- Avoid congestion
- Meet timing requirements
- Ensure the layout is efficient and error-free

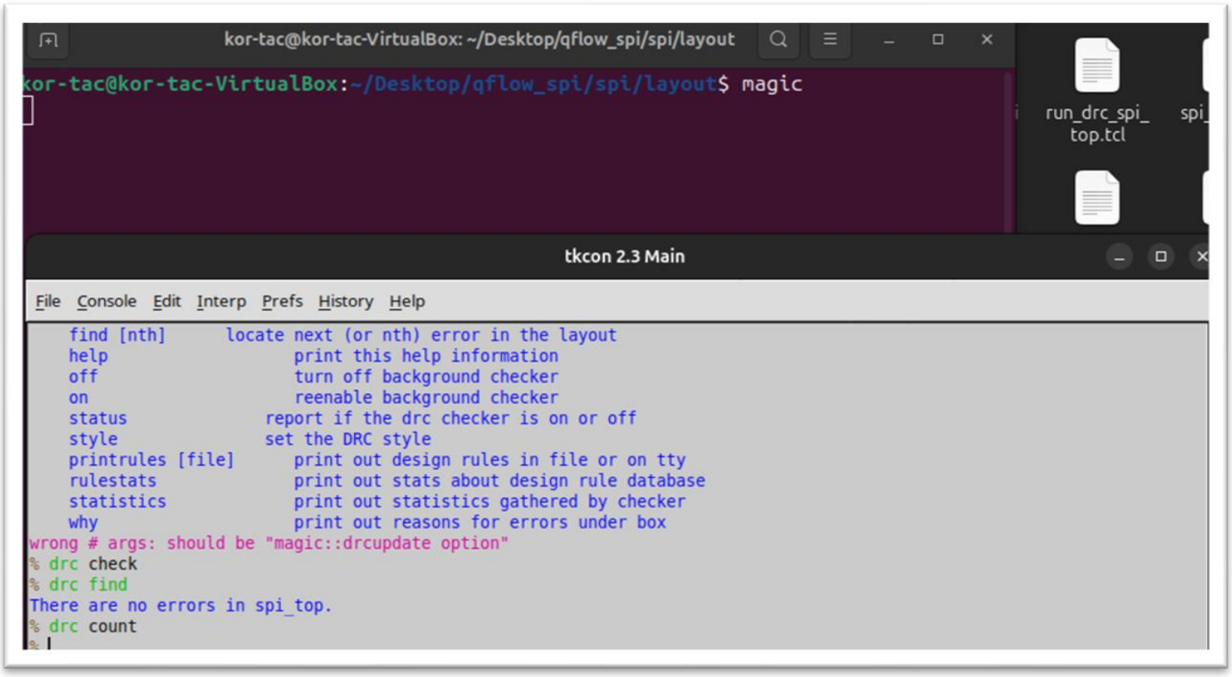
Qrouter Tool: Qflow uses a tool called Qrouter for this.

Qrouter ensures:

- Wires are routed efficiently
- The design remains within constraints
- Delays and parasitics (unwanted effects) are minimized

It’s essential for turning a logical design into a physical one that works in real hardware.

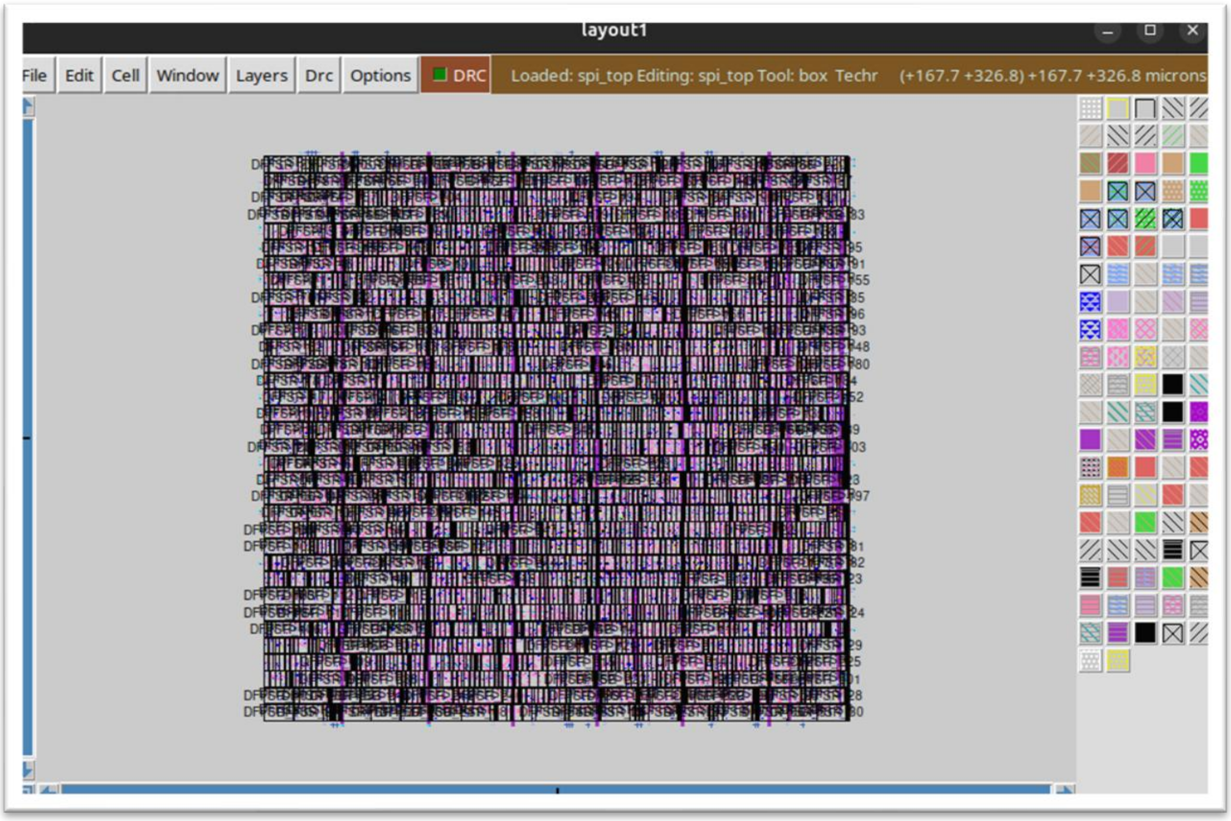
DESIGN RULE CHECKING



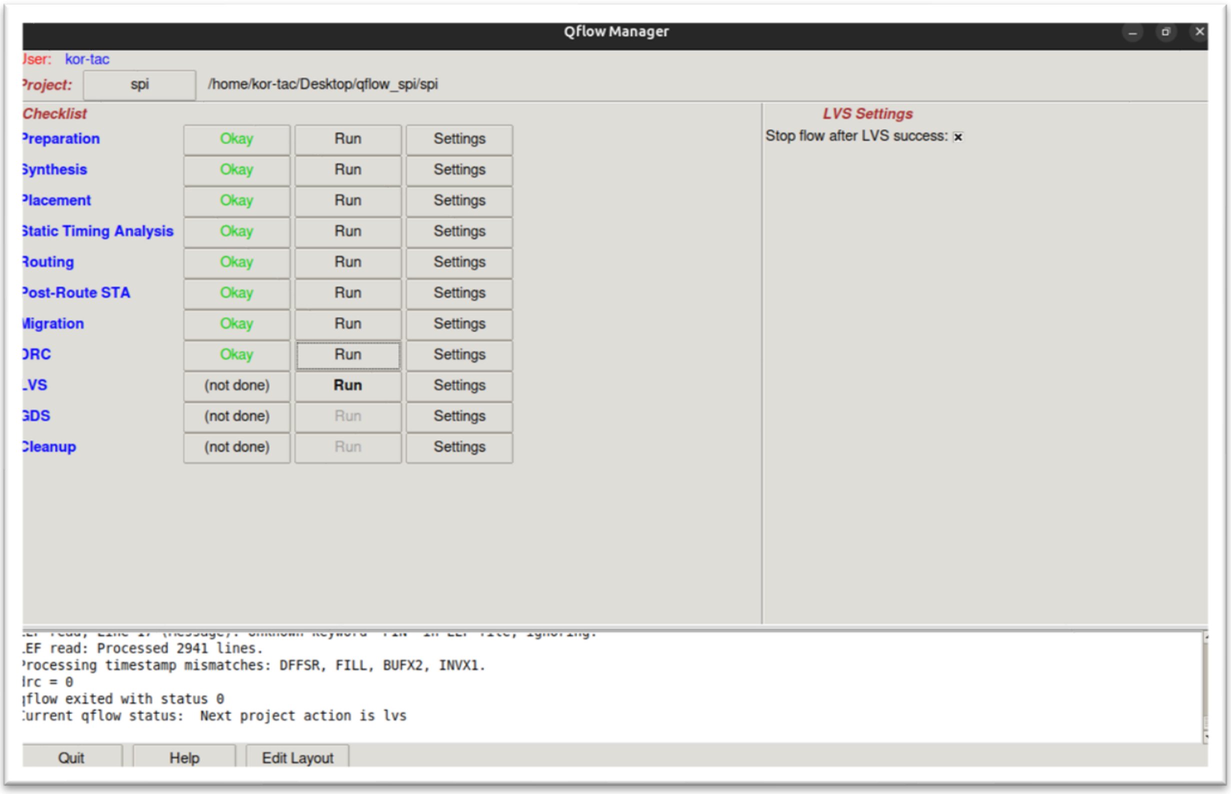
kor-tac@kor-tac-VirtualBox: ~/Desktop/qflow_spi/spi/layout
run_drc_spi_top.tcl
spi
tkcon 2.3 Main
File Console Edit Interp Prefs History Help

```
kor-tac@kor-tac-VirtualBox:~/Desktop/qflow_spi/spi/layout$ magic
```

```
    find [nth]      locate next (or nth) error in the layout
    help                   print this help information
    off                    turn off background checker
    on                     reenable background checker
    status                report if the drc checker is on or off
    style                 set the DRC style
    printrules [file]      print out design rules in file or on tty
    rulestats              print out stats about design rule database
    statistics             print out statistics gathered by checker
    why                    print out reasons for errors under box
wrong # args: should be "magic::drcupdate option"
% drc check
% drc find
There are no errors in spi_top.
% drc count
%
```



layout1
File
Edit
Cell
Window
Layers
Drc
Options
DRC
Loaded: spi_top Editing: spi_top Tool: box Techr
(+167.7 +326.8) +167.7 +326.8 microns


Qflow Manager
Jser: kor-tac
Project: spi /home/kor-tac/Desktop/qflow_spi/spi
LVS Settings
Stop flow after LVS success:
Quit
Help
Edit Layout

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	(not done)	Run	Settings
GDS	(not done)	Run	Settings
Cleanup	(not done)	Run	Settings

```
[illegible]
.EF read: Processed 2941 lines.
'rocessing timestamp mismatches: DFFSR, FILL, BUFX2, INVX1.
lrc = 0
ıflow exited with status 0
:urrent qflow status:  Next project action is lvs
```

DRC is a critical step in the FPGA design process. It checks if the physical layout of a circuit follows specific manufacturing rules to ensure it can be reliably fabricated.

It verifies:

- Wire widths
- Spacing between components
- Alignment and dimensions

Violating these rules can lead to defects or failures. DRC helps avoid costly design errors by catching issues before manufacturing.

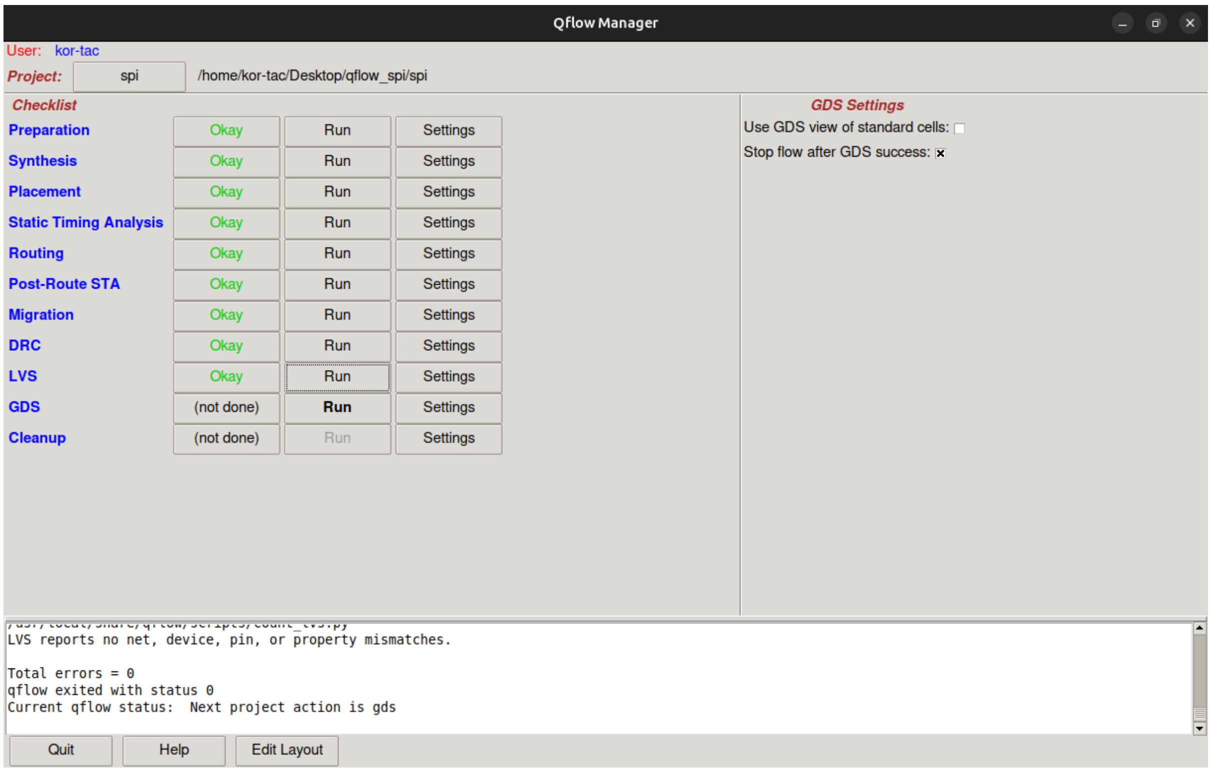
Tools Used:

- Magic is a tool used in Qflow to perform DRC.
- It visually highlights errors and helps in debugging layout violations.

DRC Error Fixing Workflow (in Qflow):

1. Load the design in Magic.
2. Use drc count to check the number of violations.
3. Use drc why in the console to see what's wrong.
4. Navigate to the error in the layout.
5. Fix the error.
6. Recheck and re-run DRC.

LAYOUT vs SCHEMATICS (LVS)


Qflow Manager

User: kor-tac

Project: spi /home/kor-tac/Desktop/qflow_spi/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: ☒

```
/usr/local/share/qflow/scripts/count_lvs.py
LVS reports no net, device, pin, or property mismatches.

Total errors = 0
qflow exited with status 0
Current qflow status:  Next project action is gds
```

LVS is a critical verification step in physical design. It ensures that the physical layout of a circuit matches its original schematic (i.e., the logical design).

Here's how it works:

- Compares netlists (lists of components and their connections) from both the layout and the schematic.
- Verifies if the physical implementation (layout) is logically equivalent to the intended design (schematic).
- Helps identify mismatches like:
    - Missing or extra components
    - Wrong connections
    - Incorrect wiring

LVS is essential because any mismatch can lead to circuit failure or fabrication issues.

Why LVS is Important:

- Ensures design integrity before manufacturing.
- Prevents costly errors.
- Helps confirm that layout tools haven’t introduced errors during design.

In the Qflow FPGA flow, LVS tools compare the layout netlist with the original design to make sure everything is connected as planned.

GDS (GRAPHIC DATA SYSTEM)

GDS is a file format used in FPGA and VLSI design that stores the final physical layout of a circuit. It acts as the blueprint for fabrication and includes data about:

- Geometry (shapes and patterns)
- Placement
- Interconnections of components

GDS ensures that:

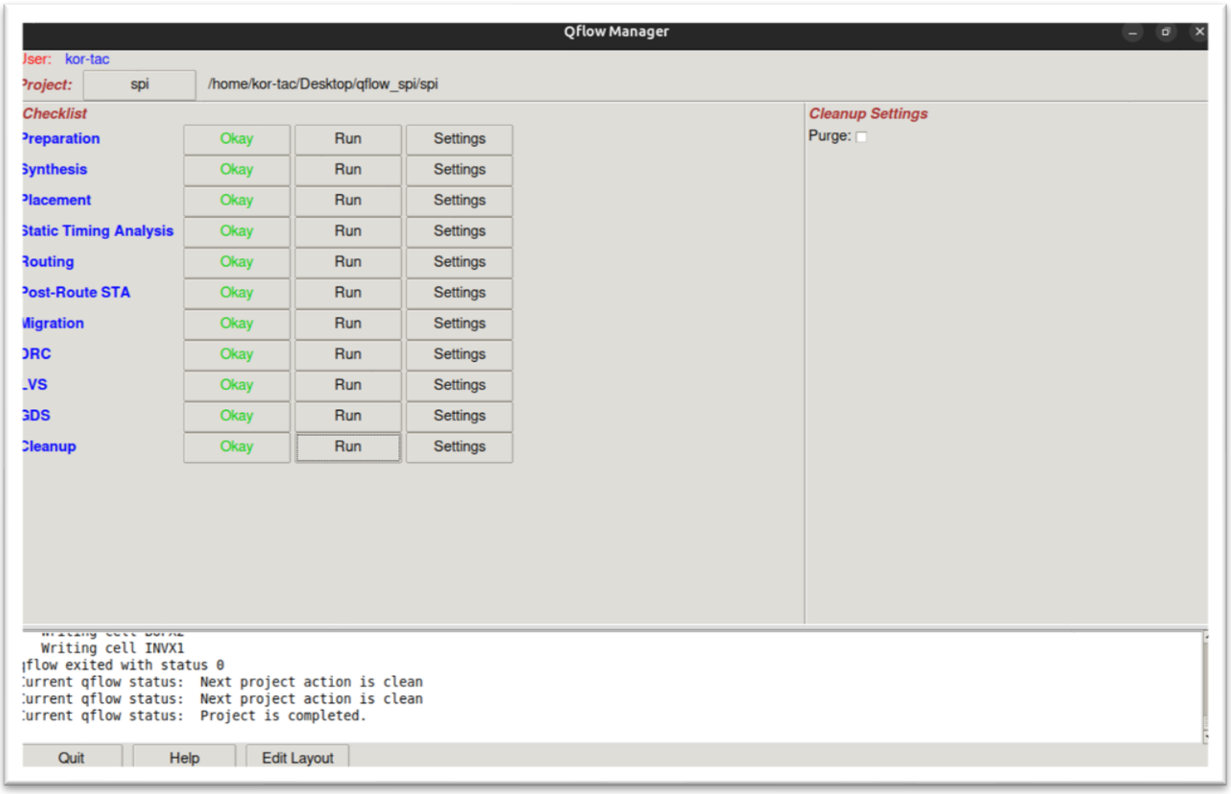
- The design can be accurately interpreted by fabrication tools
- The layout info can be easily shared across teams and tools
- Errors are minimized during chip manufacturing

In the Qflow workflow, tools like Magic generate the GDS file. This GDS file is then used to create masks during the chip manufacturing process.

Why GDS Matters:

- It's the last step before fabrication.
- Helps bridge the gap between design and manufacturing.
- Ensures the physical design matches what was logically intended.

FINAL OUTPUT VIEW


Qflow Manager

ser: kor-tac

roject: spi /home/kor-tac/Desktop/qflow_spi/spi

Checklist

reparation	Okay	Run	Settings
ynthesis	Okay	Run	Settings
lacement	Okay	Run	Settings
tatic Timing Analysis	Okay	Run	Settings
outing	Okay	Run	Settings
ost-Route STA	Okay	Run	Settings
igration	Okay	Run	Settings
RC	Okay	Run	Settings
VS	Okay	Run	Settings
DS	Okay	Run	Settings
leanup	Okay	Run	Settings

Cleanup Settings

Purge: ☐

```
  Writing cell INVX1
flow exited with status 0
urrent qflow status:  Next project action is clean
urrent qflow status:  Next project action is clean
urrent qflow status:  Project is completed.
```

This section shows the completion status of the Qflow design flow, including:

- A screenshot of the Qflow Manager showing all steps from synthesis to GDS generation.
- All essential stages like synthesis, placement, STA (Static Timing Analysis), routing, DRC (Design Rule Check), LVS (Layout vs Schematic), and GDS are marked as successfully completed.

A Project Report By:

S Steven Joshua

03.07.2025 - Chennai