Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 08, August-2023

# ASIC Design for a 32-bit RISC-V Processor

Poorvaja Harish[1], Ravishankar Holla[2]
[1]MTech student, Dept. of Electronics and Communication, RV College of Engineering, Bengaluru
[2]Assistant Professor, Dept. of Electronics and Communication, RV College of Engineering, Bengaluru

***Abstract* - Qflow is an open-source EDA (Electronic Design Automation) flow primarily focused on digital VLSI design. It provides a set of tools and scripts that facilitate various stages of the chip design process, including synthesis, placement, routing, and verification. Commercial tools like Cadence and Synopsis require expensive license purchase for chip design. QFlow's open-source nature allows designers to access and modify the underlying tools, enabling customization and experimentation. Here, in this paper, the RTL to GDSII flow is performed for a 32-bit RISC-V processor using Qflow in 180nm technology.**

***Keywords* - RISC-V, Open-source, RTL, GDSII, Qflow**

## I. INTRODUCTION

RISC-V, [11] an open-source Instruction Set Architecture (ISA), has garnered substantial attention for its innovative approach to processor design. The acronym stands for" Reduced Instruction Set Computing - Five". The defining characteristic of RISC-V is its accessibility and open availability, enabling unrestricted use, modification, and implementation by the global community. The RISC-V architecture emerges as a promising avenue due to its open nature and adaptability, contrasting with proprietary processor designs.

The emergence of RISC-V architecture distinguishes itself from traditional licensed processor designs by embracing an open-source philosophy. In contrast to many proprietary architectures prevalent in the industry, RISC-V offers an open standard that encourages collaborative innovation and allows researchers and developers to customize and experiment with processor designs in unprecedented ways. Many licensed processor architectures come with licensing fees and may limit the level of flexibility that can be achieved. The modular nature of the RISC-V instruction set architecture allows for fine tuning processors to specific applications, providing an edge in efficiency and performance optimization.

The QFlow EDA [6] flow typically includes tools such as Yosys for synthesis, Graywolf for placement, Qrouter for routing, and Magic for layout viewing and editing. These tools are often integrated into a cohesive flow through a series of scripts and configurations. QFlow aims to simplify the process of designing digital integrated circuits using open-source tools and methodologies. It is often used for small to medium-sized projects and academic purposes. QFlow's open-source nature allows designers to access and modify the underlying tools, enabling customization and experimentation. This can be particularly useful for educational purposes and research projects where understanding and optimizing the chip design process are essential goals.

This paper explores a 32-bit processor design, aiming to leverage the advantages of RISC-V architecture within the realm of open-source EDA tools. The remaining sections of the paper are organized as follows: Section II RISC-V overview; the Section III Methodology; Experimental results present the achievements in Section IV; and finally, the Section V conclusion summarizes contributions and possible future directions.

## II. RISC-V OVERVIEW

The RISC-V Execution Pipeline operates through five sequential stages [3], each contributing to the processor's seamless operation. The execution pipeline in the RISC-V architecture comprises of five key stages: IF (Instruction Fetch), ID (Instruction Decode), EX(Execute), MEM (Memory Access), and WB(Write-Back) [2].

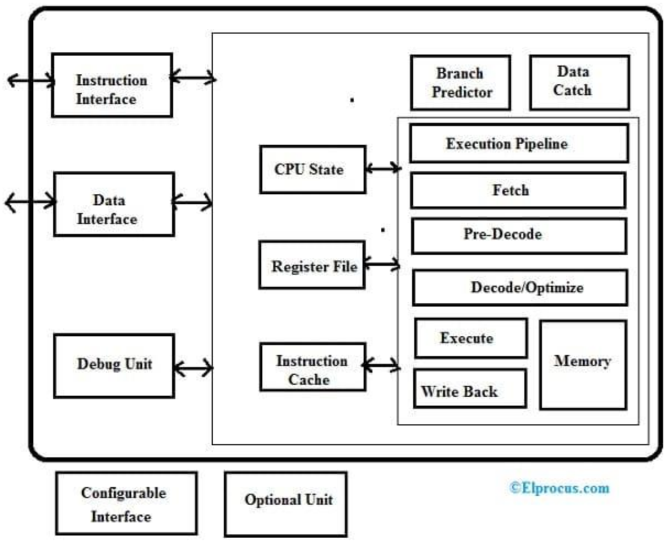


Fig. 1 Architecture of RISC-V Processor

### A. *Instruction Fetch (IF)*

During the Instruction Fetch stage, a pivotal precursor to the processor's functioning, a singular instruction is diligently retrieved from the instruction memory. The heart of this stage lies in the program counter (PC), a memory location that houses the address of the current instruction being fetched. The

PC is seamlessly updated to point to the subsequent instruction in line for execution.

### B. *Instruction Pre-Decode*

Instruction Pre-Decode stage plays a vital role in streamlining the decoding process. In scenarios where 16-bit-compressed instructions are utilized; this stage deftly decodes them into their native 32-bit counterparts. This transformation significantly simplifies subsequent stages, ensuring that instructions are processed uniformly, irrespective of their original format.

### C. *Instruction Decode (ID)*

The Instruction Decode stage forms a critical junction. Here, the processor engages with the Register File. The Register File serves as a fundamental source and destination for data during instruction execution. The bypass controls are determined. Value inside the instruction and also the opcodes are verified.

### D. *Execute (EX)*

Based on the instruction provided by the decoder, execution of required operations takes place. During the Execute stage, a wide array of tasks are performed. This includes executing operations for Arithmetic Logic Unit (ALU), Division (DIV), Multiplication (MUL) instructions, managing memory for Store or Load operations.

### E. *Memory (MEM)*

The Memory stage ensures seamless memory access by enabling data retrieval from or storage to memory through the pipeline. This inclusion significantly contributes to the pipeline's overall efficiency.

### F. *Write-Back (WB)*

The Write Back stage finalizes the execution process, writing the outcome of the Execute stage back into the Register File.

### G. *Data and Instruction Cache*

Two essential components, the Data Cache and the Instruction Cache, optimize memory access in the processor. The Data Cache enhances data memory retrieval by buffering frequently accessed memory locations. It adeptly handles various access sizes based on the XLEN value. Meanwhile, the Instruction Cache expedites instruction fetching by buffering recently fetched instructions. It operates at a cycle-by-cycle pace, fetching parcels on 16-bit boundaries. Both caches play a crucial role in enhancing the processor's overall performance.

### H. *Debug Unit and Register File*

The Debug Unit facilitates thorough examination of the CPU. The Register File, a core component, comprises 32 registers (X0 to X31). Notably, X9 is consistently set to zero. The Register File includes one write port and two read ports, allowing efficient data manipulation within the processor.

## III. METHODOLOGY

The goal of this work is to implement the complete ASIC design flow for a 32-bit RISC-V processor. The flowchart representing the steps in the flow are shown in Fig. 2. The RTL code and testbench of the processor is written in Verilog hardware description language. It consists of codes for different blocks of the processor including ALU, Control Unit, Instruction Memory, Instruction Fetch Unit, Register File. This code is simulated using iverilog. The simulation waveforms are viewed in gtkwave. After verifying the functionality comes the Preparation step where initial setup and configuration required before starting the actual ASIC design flow. This step involves several tasks to ensure that the design environment is properly organized and ready for the subsequent design stages. Some of the steps include library selection, technology file integration, design files import, constraints import, and script creation.

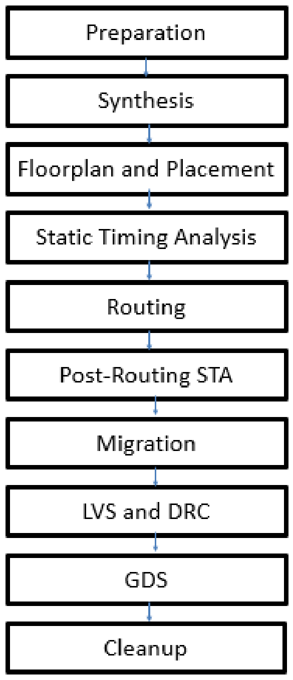
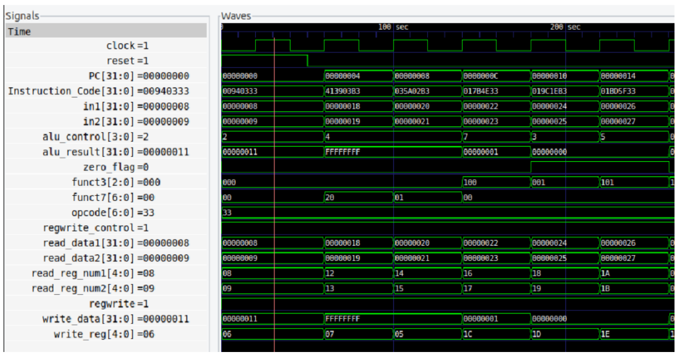


Fig. 2 VLSI Design Flow in Qflow

After preparation, the RTL description undergoes synthesis using Yosys. This phase translates the RTL code into a gate-level representation called netlist. The Floorplan and Placement step comes after this where the chip area is divided into functional blocks. The dimensions and positions of these blocks are determined and gate-level netlist is placed on the chip area allocated to each block. Tools used is Graywolf. Then STA is run for the pre-routed layout using openSTA [14] and OpenTimer tools. This gives the maximum clock frequency of the design and also if the timing requirement is met or not. Once STA is done, Routing starts. Qrouter is used to establish connections between gates while adhering to design constraints and rules. MAGIC layout editor [12] is used to edit layouts. Now we have the routed-layout for which again STA is run to check whether the clock frequency and design timing is met even after an additional load is introduced due to routing. Now checks like DRC and LVS are performed on the routed-layout. Once the design is error-free, the GDS file is generated and final cleanup is done. This completes the project.

## IV. EXPERIMENTAL RESULTS

The functional simulation of a 32-bit RISC-V processor is shown in Fig. 3 below. The open-source tool iverilog is used to compile and execute the Verilog code of the processor. The waveform is viewed in gtkwave.



```
Time
                      clock=1
                      reset=1
               PC[31:0] =00000000
Instruction_Code[31:0] =00940333
              in1[31:0] =00000008
              in2[31:0] =00000009
        alu_control[3:0] =2
        alu_result[31:0] =00000011
                  zero_flag=0
             funct3[2:0] =000
             funct7[6:0] =00
             opcode[6:0] =33
           regwrite_control=1
        read_data1[31:0] =00000008
        read_data2[31:0] =00000009
      read_reg_num1[4:0] =08
      read_reg_num2[4:0] =09
                   regwrite=1
        write_data[31:0] =00000011
           write_reg[4:0] =06
```

Fig. 3 Functional Simulation Results

Synthesis takes .sdc design constraints file and the RTL code as the input and provides the gate-level-netlist and .sdc constraint file as output. The schematic of the netlist obtained in Yosys is shown in Fig. 4.

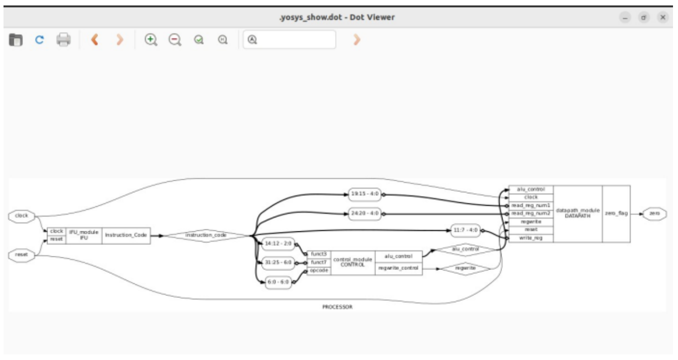
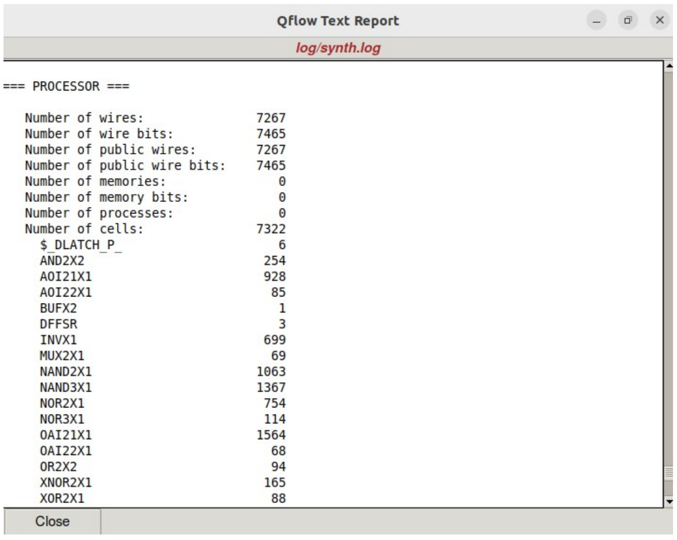


Fig. 4 Synthesis Netlist Schematic

The report generated after synthesis shows the total number of cells used in the processor design and also the number and type of individual standard cells in the design netlist. A total of 7322 cells are used in the RISC-V. Fig. 5 shows the report.

Fig. 6 shows the stages of Floorplan and Placement. In this stage the chip area is divided into different blocks and all the standard cells are placed in those blocks accordingly.

Fig. 7 shows the intermediate stage and the final placement view is shown in Fig. 8.



```
=== PROCESSOR ===

   Number of wires:               7267
   Number of wire bits:           7465
   Number of public wires:        7267
   Number of public wire bits:    7465
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:               7322
     $_DLATCH_P_                     6
     AND2X2                        254
     AOI21X1                       928
     AOI22X1                        85
     BUFX2                           1
     DFFSR                           3
     INVX1                         699
     MUX2X1                         69
     NAND2X1                      1063
     NAND3X1                      1367
     NOR2X1                        754
     NOR3X1                        114
     OAI21X1                      1564
     OAI22X1                        68
     OR2X2                          94
     XNOR2X1                       165
     XOR2X1                         88
```

Fig. 5 Synthesis Report

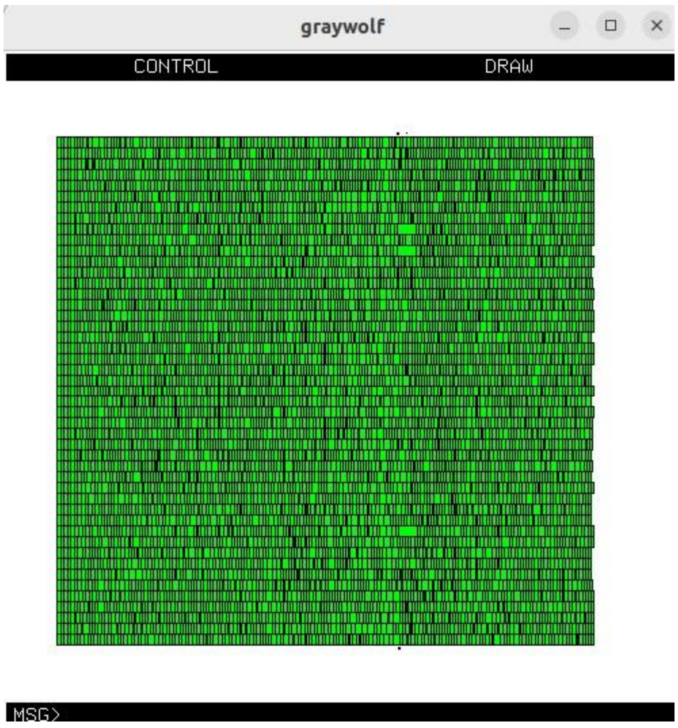


Fig. 6 Floorplan and Placement Initial Stage

The reports generated after the placement stage can be seen in Fig. 9 shows the final number of routing tracks assigned.

The layout of the processor before routing and after placement is shown in Fig. 10 and 11 in full view and zoomed view.

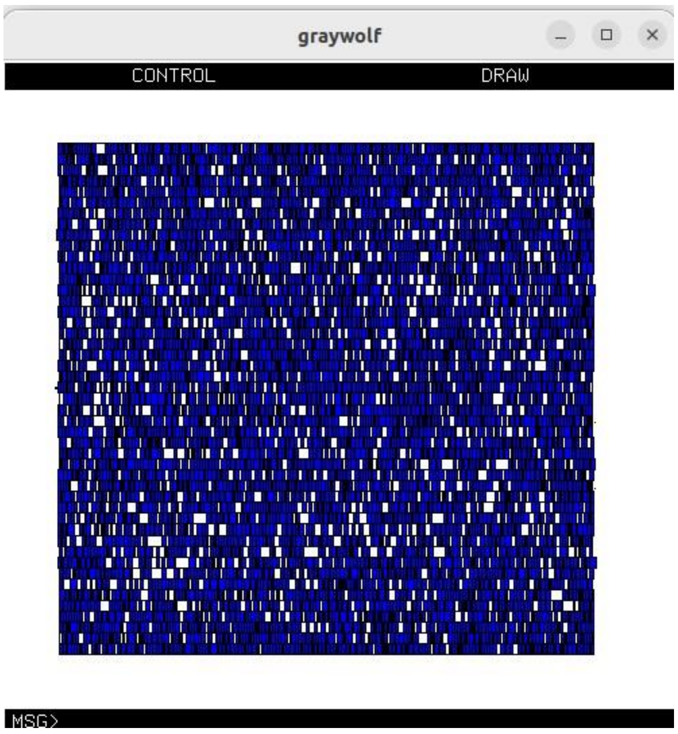


Fig. 7 Floorplan and Placement Intermediate Stage

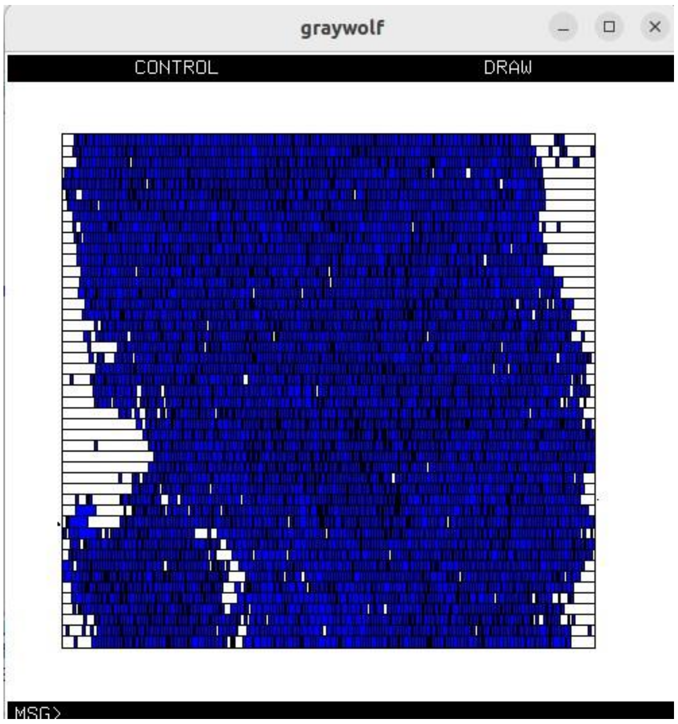
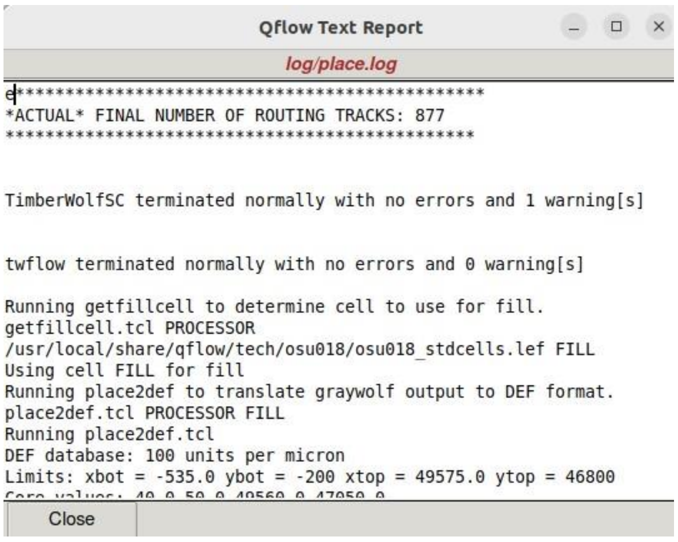


Fig. 8 Floorplan and Placement Final Stage

Once the placement of standard cells is completed, static timing analysis is run for the pre-routed layout. The pre-routed layout STA results show that the design meets the required timing and the maximum clock frequency obtained is 115.147 MHz.



```
e***********************************************
*ACTUAL* FINAL NUMBER OF ROUTING TRACKS: 877
***********************************************


TimberWolfSC terminated normally with no errors and 1 warning[s]


twflow terminated normally with no errors and 0 warning[s]

Running getfillcell to determine cell to use for fill.
getfillcell.tcl PROCESSOR
/usr/local/share/qflow/tech/osu018/osu018_stdcells.lef FILL
Using cell FILL for fill
Running place2def to translate graywolf output to DEF format.
place2def.tcl PROCESSOR FILL
Running place2def.tcl
DEF database: 100 units per micron
Limits: xbot = -535.0 ybot = -200 xtop = 49575.0 ytop = 46800
Core values: 40.0 50.0 49560.0 47050.0
```

Fig. 9 Placement Report

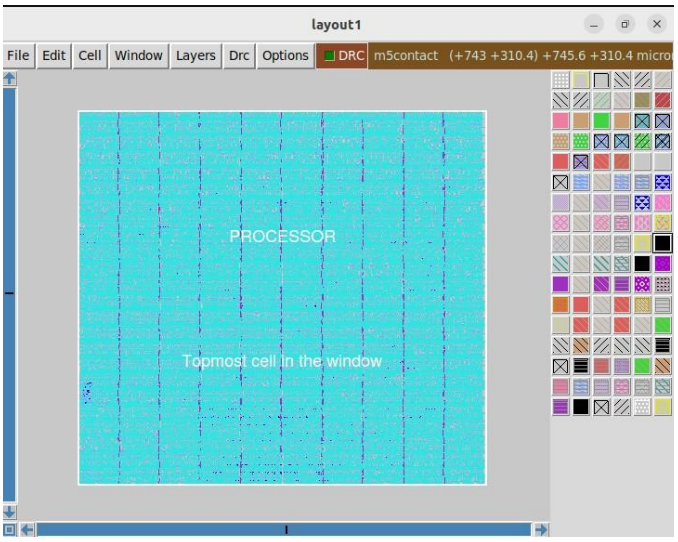

layout1
File
Edit
Cell
Window
Layers
Drc
Options
DRC
m5contact (+743 +310.4) +745.6 +310.4 micro
PROCESSOR
Topmost cell in the window

Fig. 10 Placed and un-routed layout full view

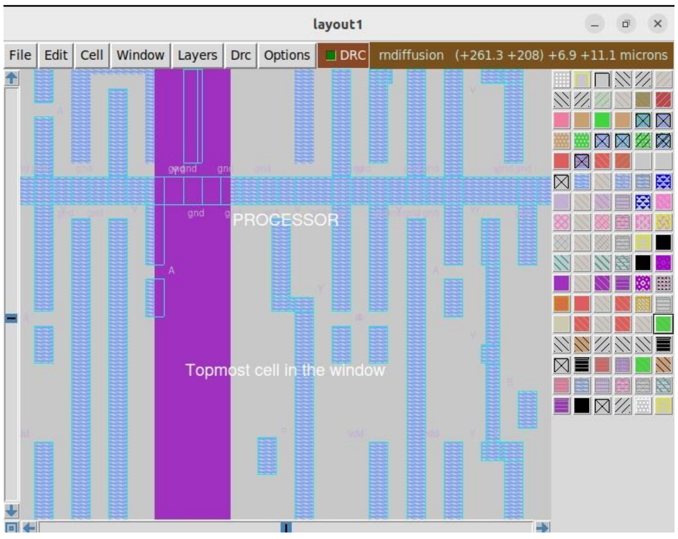
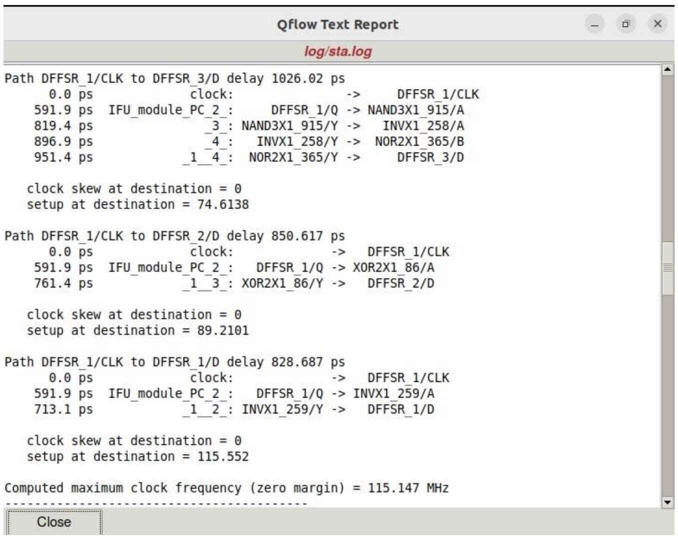

layout1
File
Edit
Cell
Window
Layers
Drc
Options
DRC
mdiffusion (+261.3 +208) +6.9 +11.1 microns
PROCESSOR
Topmost cell in the window

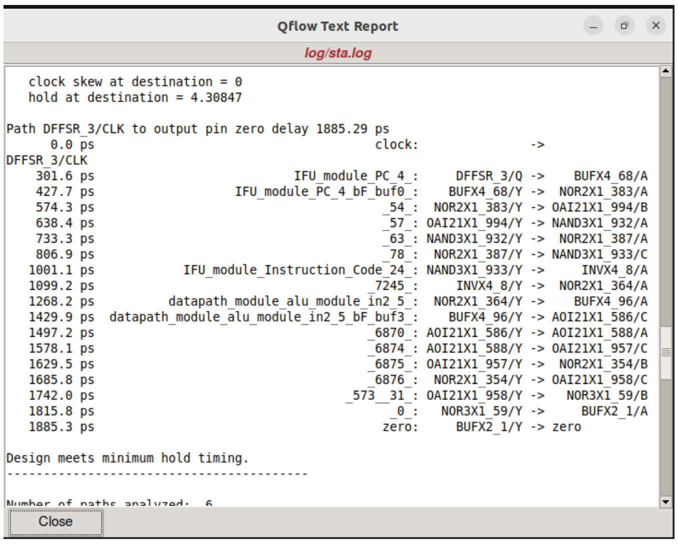
Fig. 11 Placed and un-routed layout zoomed view

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 08, August-2023



```
Path DFFSR_1/CLK to DFFSR_3/D delay 1026.02 ps
      0.0 ps               clock:               ->     DFFSR_1/CLK
    591.9 ps  IFU_module_PC_2_:     DFFSR_1/Q -> NAND3X1_915/A
    819.4 ps                 _3_: NAND3X1_915/Y ->   INVX1_258/A
    896.9 ps                 _4_:   INVX1_258/Y ->  NOR2X1_365/B
    951.4 ps              _1__4_:  NOR2X1_365/Y ->     DFFSR_3/D

   clock skew at destination = 0
   setup at destination = 74.6138

Path DFFSR_1/CLK to DFFSR_2/D delay 850.617 ps
      0.0 ps               clock:             ->   DFFSR_1/CLK
    591.9 ps  IFU_module_PC_2_:   DFFSR_1/Q -> XOR2X1_86/A
    761.4 ps              _1__3_: XOR2X1_86/Y ->   DFFSR_2/D

   clock skew at destination = 0
   setup at destination = 89.2101

Path DFFSR_1/CLK to DFFSR_1/D delay 828.687 ps
      0.0 ps               clock:             ->   DFFSR_1/CLK
    591.9 ps  IFU_module_PC_2_:   DFFSR_1/Q -> INVX1_259/A
    713.1 ps              _1__2_: INVX1_259/Y ->   DFFSR_1/D

   clock skew at destination = 0
   setup at destination = 115.552

Computed maximum clock frequency (zero margin) = 115.147 MHz
-----------------------------------------
```

Fig. 12 Pre-route STA - maximum clock frequency



```
   clock skew at destination = 0
   hold at destination = 4.30847

Path DFFSR_3/CLK to output pin zero delay 1885.29 ps
      0.0 ps                                      clock:               ->
DFFSR_3/CLK
    301.6 ps                       IFU_module_PC_4_:     DFFSR_3/Q ->    BUFX4_68/A
    427.7 ps               IFU_module_PC_4_bF_buf0_:    BUFX4_68/Y ->  NOR2X1_383/A
    574.3 ps                                    _54_:  NOR2X1_383/Y -> OAI21X1_994/B
    638.4 ps                                    _57_: OAI21X1_994/Y -> NAND3X1_932/A
    733.3 ps                                    _63_: NAND3X1_932/Y ->  NOR2X1_387/A
    806.9 ps                                    _78_:  NOR2X1_387/Y -> NAND3X1_933/C
   1001.1 ps        IFU_module_Instruction_Code_24_: NAND3X1_933/Y ->      INVX4_8/A
   1099.2 ps                                  _7245_:     INVX4_8/Y ->  NOR2X1_364/A
   1268.2 ps      datapath_module_alu_module_in2_5_:  NOR2X1_364/Y ->    BUFX4_96/A
   1429.9 ps  datapath_module_alu_module_in2_5_bF_buf3_:    BUFX4_96/Y -> AOI21X1_586/C
   1497.2 ps                                  _6870_: AOI21X1_586/Y -> AOI21X1_588/A
   1578.1 ps                                  _6874_: AOI21X1_588/Y -> OAI21X1_957/C
   1629.5 ps                                  _6875_: OAI21X1_957/Y ->  NOR2X1_354/B
   1685.8 ps                                  _6876_:  NOR2X1_354/Y -> OAI21X1_958/C
   1742.0 ps                                _573__31_: OAI21X1_958/Y ->   NOR3X1_59/B
   1815.8 ps                                     _0_:   NOR3X1_59/Y ->    BUFX2_1/A
   1885.3 ps                                    zero:     BUFX2_1/Y -> zero

Design meets minimum hold timing.
-----------------------------------------

Number of paths analyzed:  6
```

Fig. 13 Pre-route STA – timing met report

Fig. 14 shows the routing stage where the interconnections between the components is done. Qrouter is the tool that does routing in Qflow toolchain.

The report generated after routing shows that there are no routing errors in Fig. 15.

Post-layout STA results shown in Fig. 16 and 17 depict that the maximum clock frequency is 108.782 MHz MHz after routing. The reduction in frequency is due to delay variation, cross-talk and noise, routing congestion, clock skew and various parasitic effects.

The layout after routing can be compared with the layout before routing and significant changes can be seen due to interconnection.

The Fig. 18 shows the full view and the Fig. 19 shows the zoomed view of the layout post-routing stage.

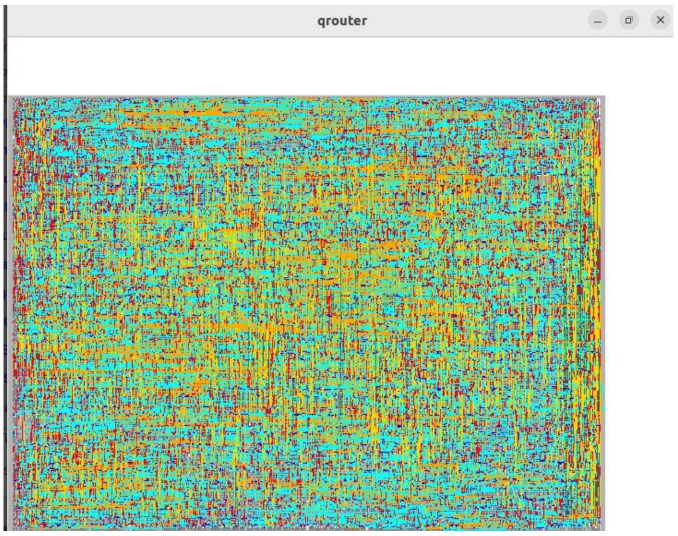
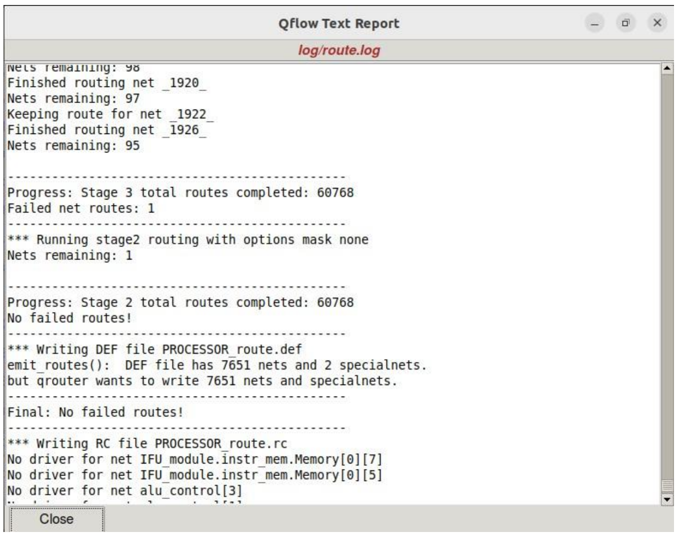
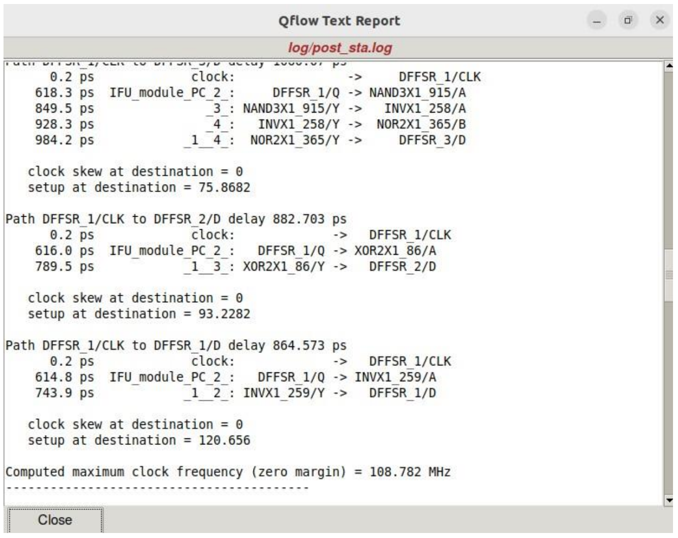


Fig. 14 Qrouter routing



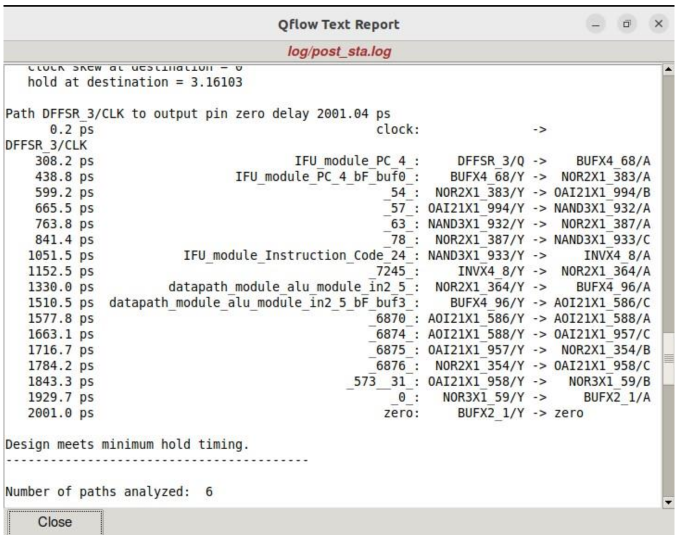
```
Nets remaining: 98
Finished routing net _1920_
Nets remaining: 97
Keeping route for net _1922_
Finished routing net _1926_
Nets remaining: 95

----------------------------------------------
Progress: Stage 3 total routes completed: 60768
Failed net routes: 1
----------------------------------------------
*** Running stage2 routing with options mask none
Nets remaining: 1

----------------------------------------------
Progress: Stage 2 total routes completed: 60768
No failed routes!
----------------------------------------------
*** Writing DEF file PROCESSOR_route.def
emit_routes():  DEF file has 7651 nets and 2 specialnets.
but qrouter wants to write 7651 nets and specialnets.
----------------------------------------------
Final: No failed routes!
----------------------------------------------
*** Writing RC file PROCESSOR_route.rc
No driver for net IFU_module.instr_mem.Memory[0][7]
No driver for net IFU_module.instr_mem.Memory[0][5]
No driver for net alu_control[3]
```

Fig. 15 Routing Report



```
      0.2 ps               clock:               ->     DFFSR_1/CLK
    618.3 ps  IFU_module_PC_2_:     DFFSR_1/Q -> NAND3X1_915/A
    849.5 ps                 _3_: NAND3X1_915/Y ->   INVX1_258/A
    928.3 ps                 _4_:   INVX1_258/Y ->  NOR2X1_365/B
    984.2 ps              _1__4_:  NOR2X1_365/Y ->     DFFSR_3/D

   clock skew at destination = 0
   setup at destination = 75.8682

Path DFFSR_1/CLK to DFFSR_2/D delay 882.703 ps
      0.2 ps               clock:             ->   DFFSR_1/CLK
    616.0 ps  IFU_module_PC_2_:   DFFSR_1/Q -> XOR2X1_86/A
    789.5 ps              _1__3_: XOR2X1_86/Y ->   DFFSR_2/D

   clock skew at destination = 0
   setup at destination = 93.2282

Path DFFSR_1/CLK to DFFSR_1/D delay 864.573 ps
      0.2 ps               clock:             ->   DFFSR_1/CLK
    614.8 ps  IFU_module_PC_2_:   DFFSR_1/Q -> INVX1_259/A
    743.9 ps              _1__2_: INVX1_259/Y ->   DFFSR_1/D

   clock skew at destination = 0
   setup at destination = 120.656

Computed maximum clock frequency (zero margin) = 108.782 MHz
-----------------------------------------
```

Fig. 16 Post-route STA – maximum clock frequency



```
   clock skew at destination = 0
   hold at destination = 3.16103

Path DFFSR_3/CLK to output pin zero delay 2001.04 ps
      0.2 ps                                       clock:             ->
DFFSR_3/CLK
    308.2 ps                        IFU_module_PC_4_:     DFFSR_3/Q ->     BUFX4_68/A
    438.8 ps                IFU_module_PC_4_bF_buf0_:    BUFX4_68/Y ->   NOR2X1_383/A
    599.2 ps                                    _54_:  NOR2X1_383/Y -> OAI21X1_994/B
    665.5 ps                                    _57_: OAI21X1_994/Y -> NAND3X1_932/A
    763.8 ps                                    _63_: NAND3X1_932/Y ->   NOR2X1_387/A
    841.4 ps                                    _78_:  NOR2X1_387/Y -> NAND3X1_933/C
   1051.5 ps           IFU_module_Instruction_Code_24_: NAND3X1_933/Y ->     INVX4_8/A
   1152.5 ps                                  _7245_:     INVX4_8/Y ->   NOR2X1_364/A
   1330.0 ps         datapath_module_alu_module_in2_5_:  NOR2X1_364/Y ->     BUFX4_96/A
   1510.5 ps  datapath_module_alu_module_in2_5_bF_buf3_:    BUFX4_96/Y -> AOI21X1_586/C
   1577.8 ps                                  _6870_: AOI21X1_586/Y -> AOI21X1_588/A
   1663.1 ps                                  _6874_: AOI21X1_588/Y -> OAI21X1_957/C
   1716.7 ps                                  _6875_: OAI21X1_957/Y ->   NOR2X1_354/B
   1784.2 ps                                  _6876_:  NOR2X1_354/Y -> OAI21X1_958/C
   1843.3 ps                               _573__31_: OAI21X1_958/Y ->    NOR3X1_59/B
   1929.7 ps                                     _0_:   NOR3X1_59/Y ->     BUFX2_1/A
   2001.0 ps                                   zero:     BUFX2_1/Y -> zero

Design meets minimum hold timing.
-----------------------------------------

Number of paths analyzed:  6
```

Fig. 17 Post-route STA – timing met report

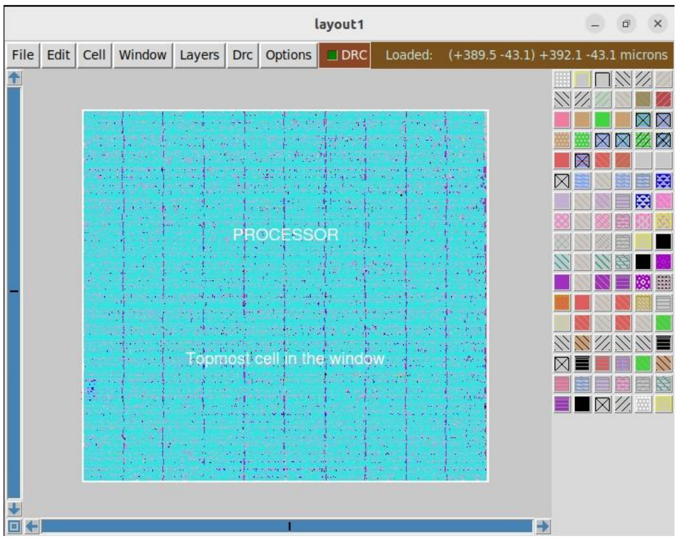

layout1
File
Edit
Cell
Window
Layers
Drc
Options
DRC
Loaded:
(+389.5 -43.1) +392.1 -43.1 microns
PROCESSOR
Topmost cell in the window

Fig. 18 Routed layout full view

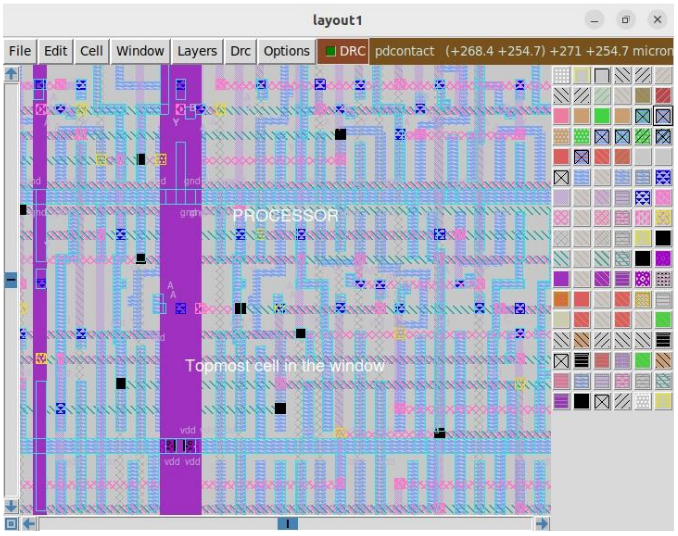
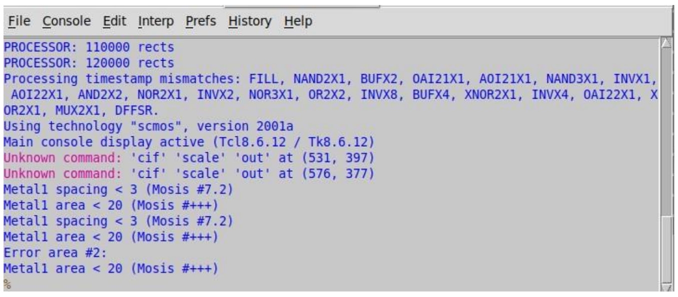


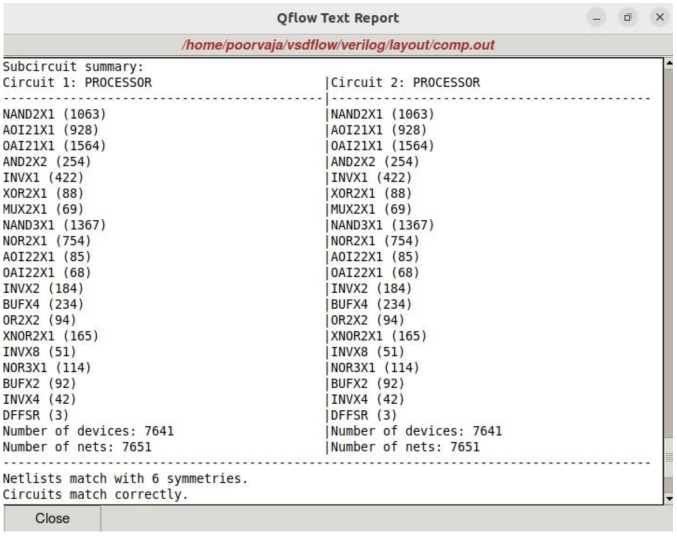
Fig. 19 Routed layout zoomed view



```
PROCESSOR: 110000 rects
PROCESSOR: 120000 rects
Processing timestamp mismatches: FILL, NAND2X1, BUFX2, OAI21X1, AOI21X1, NAND3X1, INVX1,
 AOI22X1, AND2X2, NOR2X1, INVX2, NOR3X1, OR2X2, INVX8, BUFX4, XNOR2X1, INVX4, OAI22X1, X
OR2X1, MUX2X1, DFFSR.
Using technology "scmos", version 2001a
Main console display active (Tcl8.6.12 / Tk8.6.12)
Unknown command: 'cif' 'scale' 'out' at (531, 397)
Unknown command: 'cif' 'scale' 'out' at (576, 377)
Metal1 spacing < 3 (Mosis #7.2)
Metal1 area < 20 (Mosis #+++)
Metal1 spacing < 3 (Mosis #7.2)
Metal1 area < 20 (Mosis #+++)
Error area #2:
Metal1 area < 20 (Mosis #+++)
%
```

Fig. 20 DRC Errors encountered

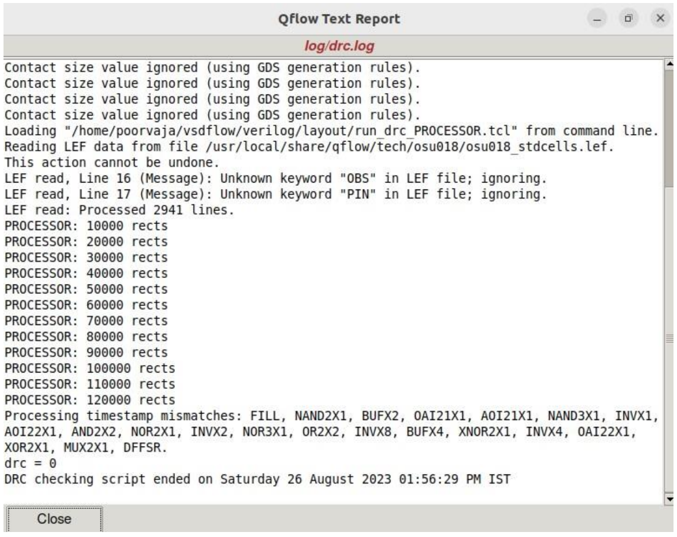
The DRC (Design Rule Check) and LVS (Layout vs Schematic) checks were done post-route. Initially few DRC errors were encountered which are shown in Fig. 20. These errors were then solved by editing the layout using MAGIC layout editor tool.



```
Subcircuit summary:
Circuit 1: PROCESSOR                        |Circuit 2: PROCESSOR
--------------------------------------------|-------------------------------------
NAND2X1 (1063)                              |NAND2X1 (1063)
AOI21X1 (928)                               |AOI21X1 (928)
OAI21X1 (1564)                              |OAI21X1 (1564)
AND2X2 (254)                                |AND2X2 (254)
INVX1 (422)                                 |INVX1 (422)
XOR2X1 (88)                                 |XOR2X1 (88)
MUX2X1 (69)                                 |MUX2X1 (69)
NAND3X1 (1367)                              |NAND3X1 (1367)
NOR2X1 (754)                                |NOR2X1 (754)
AOI22X1 (85)                                |AOI22X1 (85)
OAI22X1 (68)                                |OAI22X1 (68)
INVX2 (184)                                 |INVX2 (184)
BUFX4 (234)                                 |BUFX4 (234)
OR2X2 (94)                                  |OR2X2 (94)
XNOR2X1 (165)                               |XNOR2X1 (165)
INVX8 (51)                                  |INVX8 (51)
NOR3X1 (114)                                |NOR3X1 (114)
BUFX2 (92)                                  |BUFX2 (92)
INVX4 (42)                                  |INVX4 (42)
DFFSR (3)                                   |DFFSR (3)
Number of devices: 7641                     |Number of devices: 7641
Number of nets: 7651                        |Number of nets: 7651
---------------------------------------------------------------------------------
Netlists match with 6 symmetries.
Circuits match correctly.
```

Fig. 21 LVS Report

The reports showing DRC and LVS checks are shown in Fig. 21 and 22 respectively.



```
Contact size value ignored (using GDS generation rules).
Contact size value ignored (using GDS generation rules).
Contact size value ignored (using GDS generation rules).
Contact size value ignored (using GDS generation rules).
Loading "/home/poorvaja/vsdflow/verilog/layout/run_drc_PROCESSOR.tcl" from command line.
Reading LEF data from file /usr/local/share/qflow/tech/osu018/osu018_stdcells.lef.
This action cannot be undone.
LEF read, Line 16 (Message): Unknown keyword "OBS" in LEF file; ignoring.
LEF read, Line 17 (Message): Unknown keyword "PIN" in LEF file; ignoring.
LEF read: Processed 2941 lines.
PROCESSOR: 10000 rects
PROCESSOR: 20000 rects
PROCESSOR: 30000 rects
PROCESSOR: 40000 rects
PROCESSOR: 50000 rects
PROCESSOR: 60000 rects
PROCESSOR: 70000 rects
PROCESSOR: 80000 rects
PROCESSOR: 90000 rects
PROCESSOR: 100000 rects
PROCESSOR: 110000 rects
PROCESSOR: 120000 rects
Processing timestamp mismatches: FILL, NAND2X1, BUFX2, OAI21X1, AOI21X1, NAND3X1, INVX1,
AOI22X1, AND2X2, NOR2X1, INVX2, NOR3X1, OR2X2, INVX8, BUFX4, XNOR2X1, INVX4, OAI22X1,
XOR2X1, MUX2X1, DFFSR.
drc = 0
DRC checking script ended on Saturday 26 August 2023 01:56:29 PM IST
```

Fig. 22 DRC no error Report

DRC and LVS are clean, so the final GDSII file is generated. The final GDS layout is shown in Fig. 23 in full view and Fig. 24 in zoomed view.

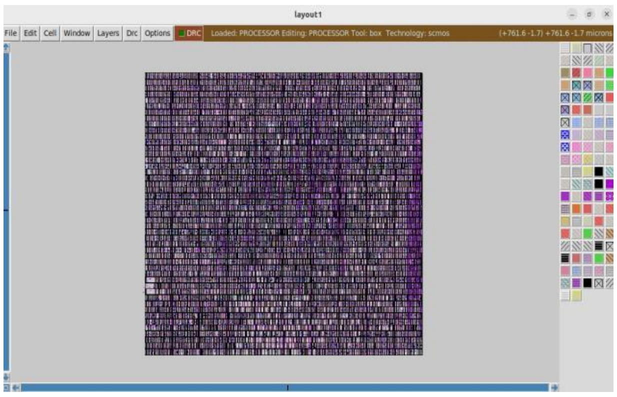

layout1
File Edit Cell Window Layers Drc Options DRC Loaded: PROCESSOR Editing: PROCESSOR Tool: box Technology: scmos
(+761.6 -1.7) +761.6 -1.7 microns

Fig. 23 GDS Layout full view

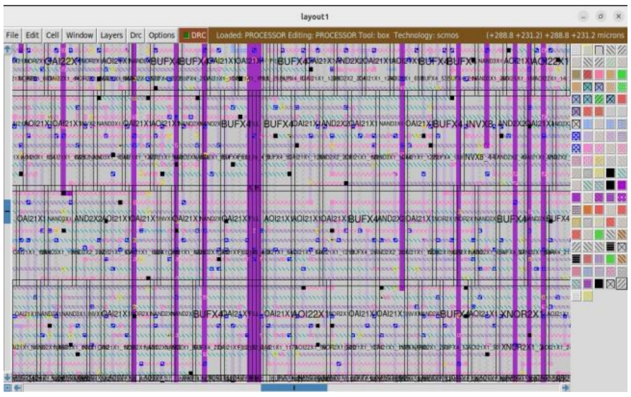
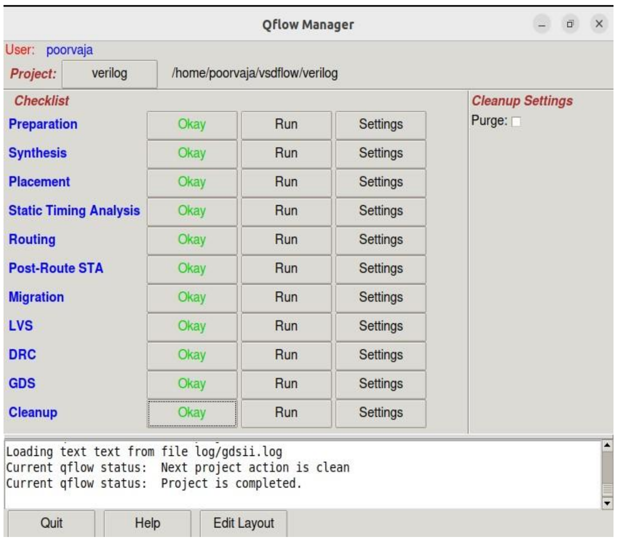

layout1
File Edit Cell Window Layers Drc Options DRC Loaded: PROCESSOR Editing: PROCESSOR Tool: box Technology: scmos
(+288.8 +231.2) +288.8 +231.2 microns

Fig. 24 GDS Layout zoomed view

## V. ANALYSIS OF THE RESULT



Qflow Manager

User: poorvaja

Project: verilog /home/poorvaja/vsdflow/verilog

| Checklist | | | | Cleanup Settings |
|---|---|---|---|---|
| Preparation | Okay | Run | Settings | Purge: ☐ |
| 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 | |
| LVS | Okay | Run | Settings | |
| DRC | Okay | Run | Settings | |
| GDS | Okay | Run | Settings | |
| Cleanup | Okay | Run | Settings | |

```
Loading text text from file log/gdsii.log
Current qflow status:  Next project action is clean
Current qflow status:  Project is completed.
```

Fig. 25 Qflow GUI

The analysis of the RTL to GDSII flow for the implementation of a 32-bit RISC-V processor using Qflow provides valuable insights into the design process, challenges faced, and achieved outcomes. The design metrics obtained after completing the entire RTL to GDSII flow demonstrate the successful transformation of the processor's RTL description into a physical layout. Fig. 16 shows the Qflow GUI (Graphical User Interface) depicting the successful completion of the project.

## VI. CONCLUSION

This work delved into the ASIC Design of a 32-bit RISC-V processor design, harnessing the capabilities of open-source EDA tools. The successful completion of the RTL to GDSII flow for the 32-bit RISC-V processor using Qflow underscores the viability and effectiveness of the open-source EDA tools in modern VLSI design. The achieved outcomes validate the effectiveness of the design methodologies employed in the flow and demonstrate the collaborative effort between design, synthesis, placement, routing, and physical verification stages. While the current study successfully implements the 32-bit RISC-V processor using Qflow, several avenues for future research and enhancement emerge. Further research could focus on exploring advanced optimization strategies to fine-tune the RTL to GDSII flow for 32-bit RISC-V processor

## REFERENCES

[1] S Nikhil Kumar Reddy, Shashank Viswanath Hosmath, Sharanakumar , Sandeep , Vinay B K, ”Implementation of RISC-V SoC from RTL to GDS flow using Open-Source Tools”, Ijraset Journal For Research in Applied Science and Engineering Technology, Volume 10 Issue VI June 2022, doi: https://doi.org/10.22214/ijraset.2022.44249

[2] J. -Y. Lai, C. -A. Chen, S. -L. Chen and C. -Y. Su, ”Implement 32-bit RISC-V Architecture Processor using Verilog HDL,” 2021 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), Hualien City, Taiwan, 2021, pp. 1-2, doi: 10.1109/ISPACS51563.2021.9651130.

[3] G. Kanase and N. M, ”ASIC Design of a 32-bit Low Power RISC-V based System Core for Medical Applications,” 2021 6th International Conference on Communication and Electronics Systems (ICCES), Coimbatre, India, 2021, pp. 1-5, doi: 10.1109/ICCES51350.2021.9489067.

[4] D. Acharya and U. S. Mehta, ”Performance Analysis of RTL to GDSII Flow in Opensource Tool Qflow and Commercial Tool Cadence Encounter for Synchronous FIFO,” 2022 IEEE International Conference of Electron Devices Society Kolkata Chapter (EDKCON), Kolkata, India, 2022, pp. 199-204, doi: 10.1109/EDKCON56221.2022.10032906.

[5] G. Kanase and K. B. Sowmya, ”Physical Implementation of Shift Register with respect to Timing and Dynamic Drop,” 2020 5th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2020, pp. 120-124, doi: 10.1109/ICCES48766.2020.9137916.

[6] K. P. Ghosh and A. K. Ghosh, ”Technology mediated tutorial on RISC-V CPU core implementation and sign-off using revolutionary EDA management system (EMS) — VSDFLOW,” 2018 China Semiconductor Technology International Conference (CSTIC), Shanghai, China, 2018, pp. 1-3, doi: 10.1109/CSTIC.2018.8369332.

[7] S. Hesham, M. Shalan, M. W. El-Kharashi and M. Dessouky, ”Digital ASIC Implementation of RISC-V: OpenLane and Commercial Approaches in Comparison,” 2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), Lansing, MI, USA, 2021, pp. 498- 502, doi: 10.1109/MWSCAS47672.2021.9531753.

[8] Neha Deshpande, Sowmya K B, ”A Review on ASIC Flow Employing EDA Tools by Synopsys,” SSRG International Journal of VLSI & Signal Processing, vol. 7, no. 1, pp. 15-19, 2020. Crossref, https://doi.org/10.14445/23942584/IJVSP-V7I1P104

[9] S. Gayathri and T. C. Taranath, ”RTL synthesis of case study using design compiler,” 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), Mysuru, 2017, pp. 1-7.

[10] S. Sreevidya, R. Holla and R. Raghu, ”Low Power Physical Design and Verification in 16nm FinFET Technology,” 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2019, pp. 936-940.

[11] Waterman, A., and K. Asanovic. "The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Document Version 2.2. CS Division, EECS Department." University of California, Berkeley (2017)

[12] "Magic VLSI Layout Tool," https://github.com/RTimothyEdwards/magic, 2020.

[13] Ahmed Alaa Ghazy and Mohamed Shalan, "OpenLane: The OpenSource Digital ASIC Implementation Flow", 2020.

[14] "OpenSTA," https://github.com/The-OpenROAD-Project/OpenSTA, 2020.

[15] Neha Deshpande, Sowmya K B, 2020, A Review on ASIC Synthesis Flow Employing Two Industry Standard Tools, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) ICEECT – 2020 (Volume 8 – Issue 17).

[16] J. Lu and B. Taskin, "From RTL to GDSII: An ASIC design course development using Synopsys® University Program," 2011 IEEE International Conference on Microelectronic Systems Education, San Diego, CA, USA, 2011, pp. 72-75.

[17] M. Shalan and T. Edwards, "Building OpenLANE: A 130nm OpenROAD-based Tapeout- Proven Flow : Invited Paper," 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), San Diego, CA, USA, 2020, pp. 1-6