EECS 4612 - Digital VLSI

Term Project: Design of 32-bit ALU

Abdulbasit Ali

Introduction

Modern digital systems rely heavily on the Arithmetic Logic Unit (ALU), a core computational component responsible for executing arithmetic, logic, and shift operations. As the backbone of processors, microcontrollers, and custom ASICs, the ALU plays a vital role in determining system performance, functionality, and scalability. Developing a robust, modular, and physically realizable ALU is therefore essential in contemporary VLSI and system-on-chip (SoC) design workflows.

This project focuses on the complete design, verification, and physical implementation of a scalable ALU system, beginning with a fully functional 1-bit slice and extending it hierarchically into a 32-bit datapath. The 1-bit ALU was designed to support a comprehensive set of operations, including:

- **Arithmetic operations:** transfer, addition, addition with carry, subtraction, subtraction with borrow, increment, and decrement.
- **Logic operations:** AND, OR, XOR, and NOT.
- **Shift operations:** logical shift left and logical shift right.

The initial 1-bit unit was implemented using structural Verilog and thoroughly verified through self-checking testbenches. Once validated, the slice was instantiated thirty-two times to construct two complete 32-bit ALU variants:

1. A **structural modular design** composed of interconnected 1-bit slices.
2. A **behavioral RTL design** implementing the same functionality at a higher abstraction level.

Both implementations underwent full functional verification followed by synthesis, place-and-route, and physical design checks (DRC and LVS) using the Cadence Virtuoso and Innovus toolchains. This ensured that the final layouts adhered to foundry design rules and were ready for fabrication.

Post-layout timing and area analysis revealed key differences between the two methodologies. The behavioral ALU achieved the highest performance, reaching approximately **16.39 MHz**, compared to **11.61 MHz** for the structural modular design. The behavioral version also occupied substantially less silicon area - approximately **975.726 µm²**, versus **1451.106 µm²** for the structural implementation. These improvements are attributed to cleaner routing, reduced logic depth, and greater synthesis optimization achievable in behavioral RTL descriptions.

Given its superior timing and area efficiency, the behavioral ALU was selected for full-chip integration. The design was incorporated into a top-level layout containing a pre-designed pad frame, with all necessary VDD, VSS, and signal pad connections routed across multiple metal layers. The resulting layout was exported in GDSII format, completing a fabrication-ready physical design.

Overall, this project demonstrates the development of a hierarchical ALU system from bit-level logic to chip-level implementation. It highlights the full digital design workflow, including modular design, verification, synthesis, physical implementation, and layout integration, and illustrates practical considerations such as timing closure, area optimization, and pad frame connectivity. The successful realization of the 32-bit ALU emphasizes both technical proficiency and readiness for industry-standard VLSI development processes.

Objectives

This project is divided into 2 main parts. The first is targeted towards the ASIC flow for a 1-bit ALU using a modular approach. After the full flow is completed, we move onto the second stage.

The second stage is creating a 32-bit ALU using the modular approach which takes the 1 bit-alu and replicates it 32 times. The second approach is the behavioral design where it is simply just a 32-bit wide ALU. The more efficient chip is then selected to become the final design, after connecting it to a padframe given on eClass. The full ASIC flow for this ALU is seen below. With the way I have structured this report, I start by creating the 1-bit ALU, followed by both 32-bit ALUs.

Part 1: 1 Bit ALU

1-bit ALU Elements

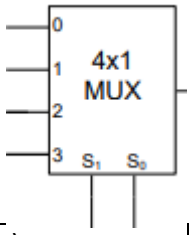
According to the manual given to us, the required components of this ALU included the following:

- 1-bit arithmetic circuit
    - 4x1 Mux
    - Full adder
- 1 bit logic circuit
    - 4x1 Mux
- MUX

With all those things combined, you can then build the ALU. The approach I took was to have a modular structure. I created a top-level file for the 1-bit ALU, with separate instantiations for the 1-bit arithmetic circuit (consisting of the mux and FA), along with the 1-bit logic circuit and another 4-1 mux. Designing these components and verifying them is critical to having this entire project function. Without a working 1-bit ALU, you won't be able to create any of the 32-bit ALUs since these sub-modules are used later. Below is the full flow.

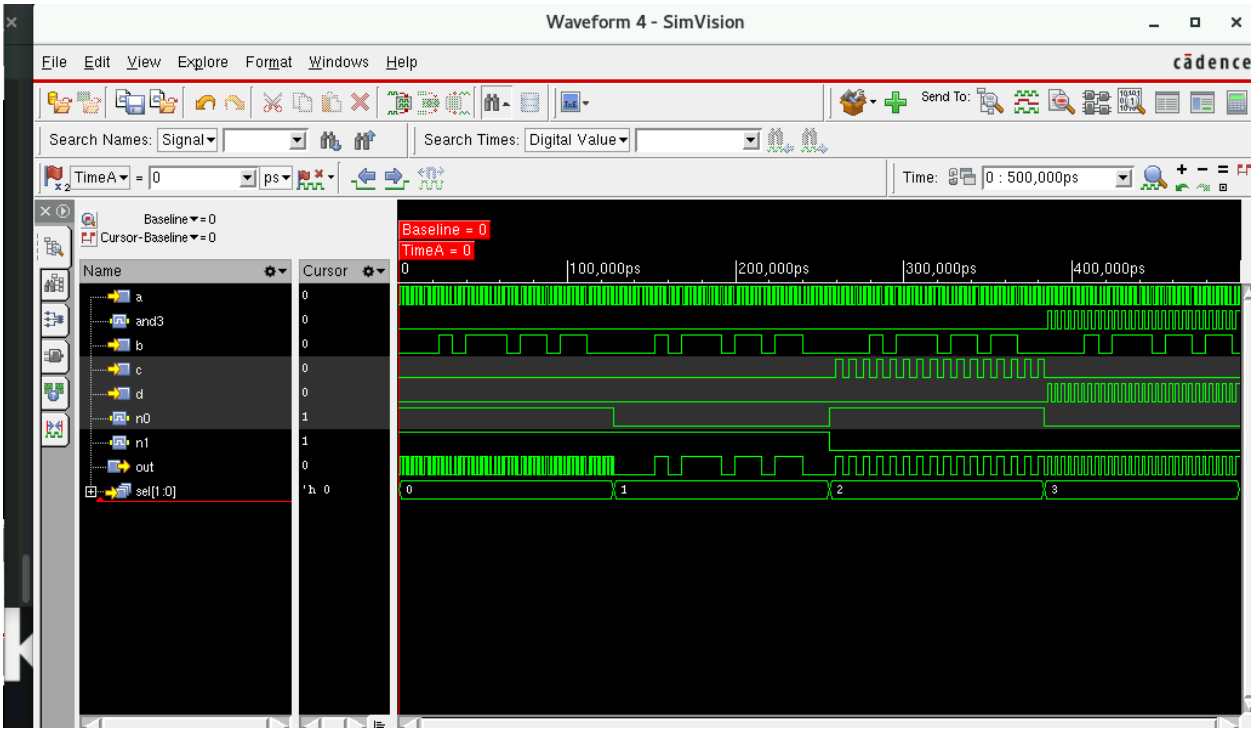
4-1 Mux Design:

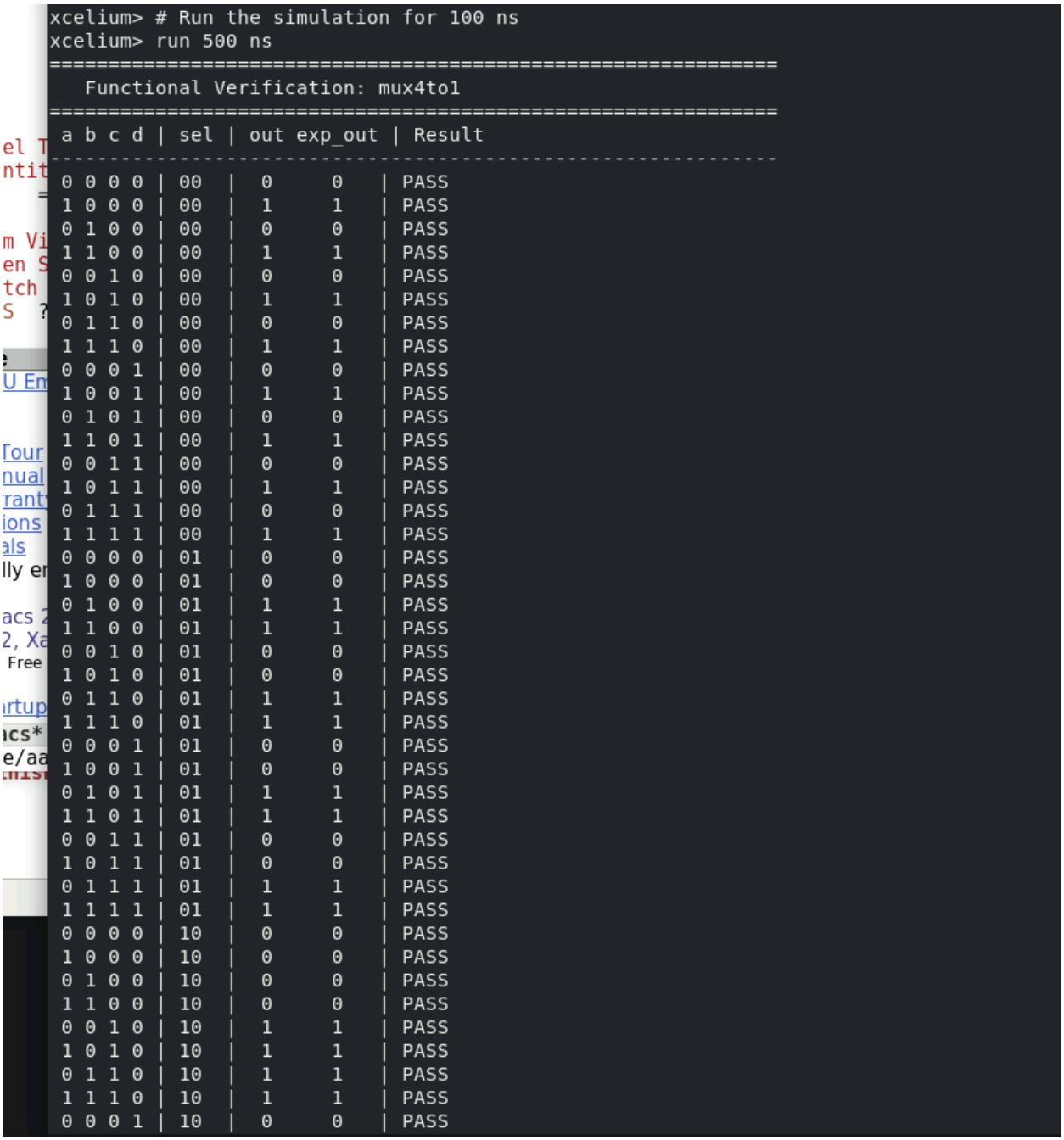
This MUX is used to enable the selection between data inputs based on other signals. The mux is implements using AND OR NOT gates. There are 2 control signals S0 and S1. The table below shows which signals result in which outputs.



S0	S1	X (input)	Y (output)
0	0	A	A
1	0	B	B
0	1	C	C
1	1	D	D

Simulated Waveform for Mux:





```
xcelium> # Run the simulation for 100 ns
xcelium> run 500 ns
================================================================
   Functional Verification: mux4to1
================================================================
 a b c d | sel | out exp_out | Result
----------------------------------------------------------------
 0 0 0 0 | 00  |  0     0   | PASS
 1 0 0 0 | 00  |  1     1   | PASS
 0 1 0 0 | 00  |  0     0   | PASS
 1 1 0 0 | 00  |  1     1   | PASS
 0 0 1 0 | 00  |  0     0   | PASS
 1 0 1 0 | 00  |  1     1   | PASS
 0 1 1 0 | 00  |  0     0   | PASS
 1 1 1 0 | 00  |  1     1   | PASS
 0 0 0 1 | 00  |  0     0   | PASS
 1 0 0 1 | 00  |  1     1   | PASS
 0 1 0 1 | 00  |  0     0   | PASS
 1 1 0 1 | 00  |  1     1   | PASS
 0 0 1 1 | 00  |  0     0   | PASS
 1 0 1 1 | 00  |  1     1   | PASS
 0 1 1 1 | 00  |  0     0   | PASS
 1 1 1 1 | 00  |  1     1   | PASS
 0 0 0 0 | 01  |  0     0   | PASS
 1 0 0 0 | 01  |  0     0   | PASS
 0 1 0 0 | 01  |  1     1   | PASS
 1 1 0 0 | 01  |  1     1   | PASS
 0 0 1 0 | 01  |  0     0   | PASS
 1 0 1 0 | 01  |  0     0   | PASS
 0 1 1 0 | 01  |  1     1   | PASS
 1 1 1 0 | 01  |  1     1   | PASS
 0 0 0 1 | 01  |  0     0   | PASS
 1 0 0 1 | 01  |  0     0   | PASS
 0 1 0 1 | 01  |  1     1   | PASS
 1 1 0 1 | 01  |  1     1   | PASS
 0 0 1 1 | 01  |  0     0   | PASS
 1 0 1 1 | 01  |  0     0   | PASS
 0 1 1 1 | 01  |  1     1   | PASS
 1 1 1 1 | 01  |  1     1   | PASS
 0 0 0 0 | 10  |  0     0   | PASS
 1 0 0 0 | 10  |  0     0   | PASS
 0 1 0 0 | 10  |  0     0   | PASS
 1 1 0 0 | 10  |  0     0   | PASS
 0 0 1 0 | 10  |  1     1   | PASS
 1 0 1 0 | 10  |  1     1   | PASS
 0 1 1 0 | 10  |  1     1   | PASS
 1 1 1 0 | 10  |  1     1   | PASS
 0 0 0 1 | 10  |  0     0   | PASS
```

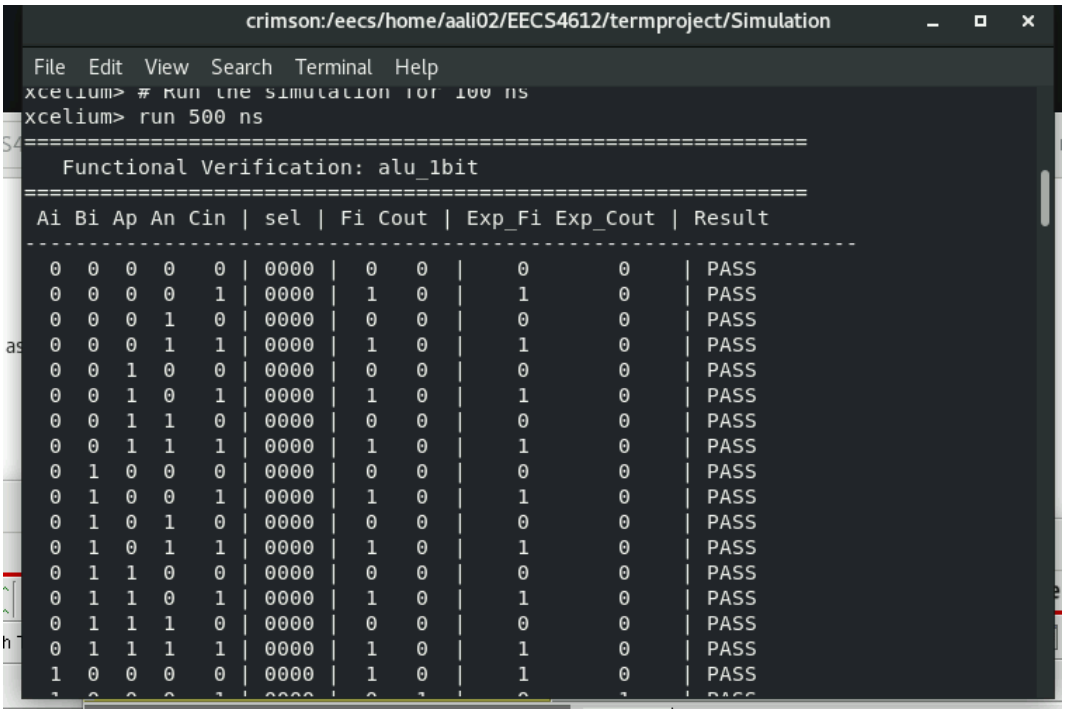
Table of MUX verification.

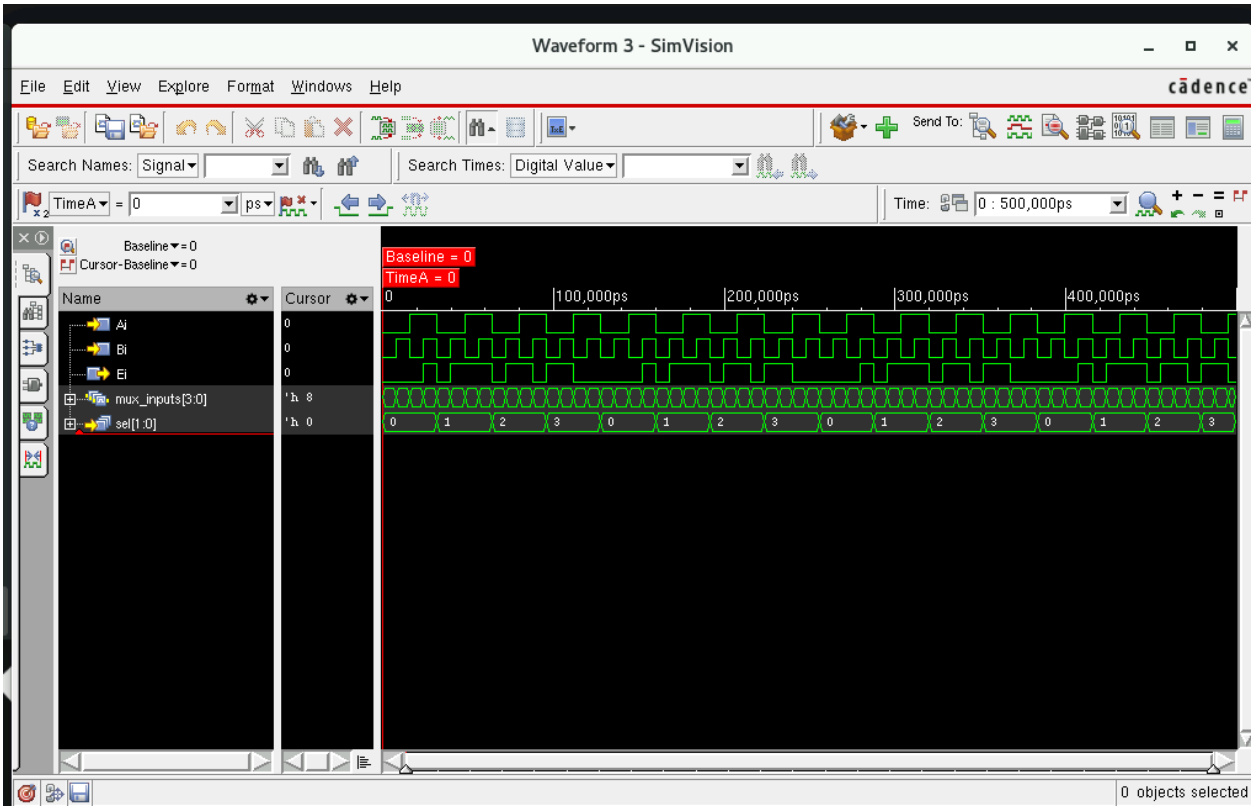
Logic Unit:

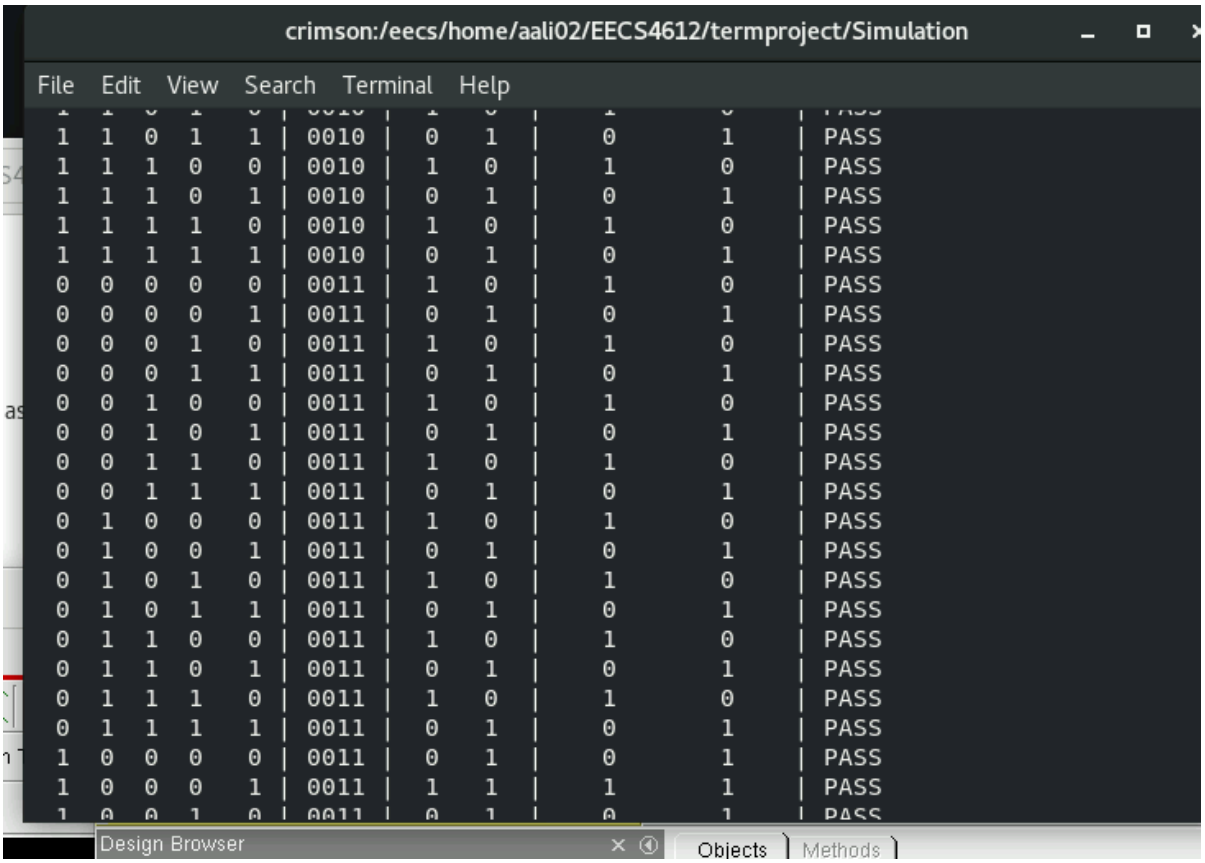
The logic unit uses a 4-1 Mux as well as some basic logic gates AND OR XOR NOT. The control is based on control signals S1 S0.

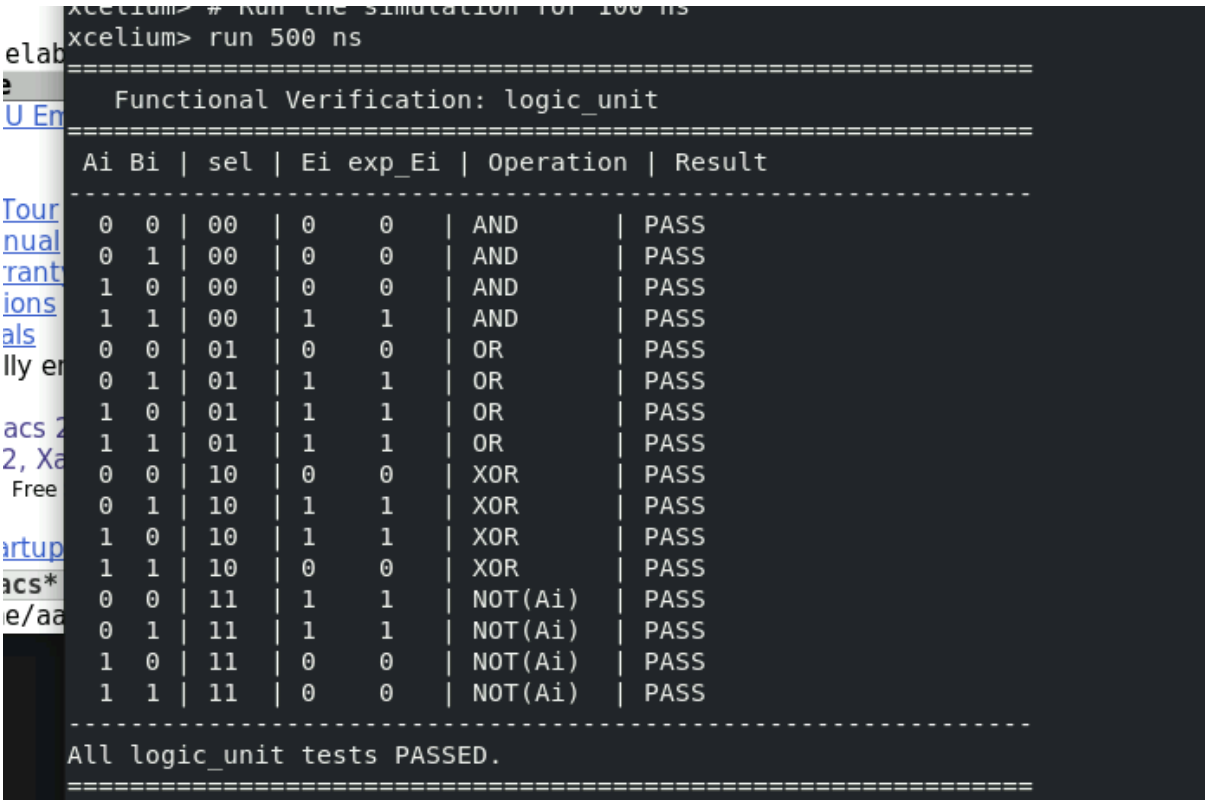
A truth table below shows the signals Ai and Bi with the output Ei. I also developed a testbench with some tests which tested all possible combinations. A truth table explaining it all is below as

well as a small example of the testbench. The truth table can be extrapolated for the whole 1-bit ALU.



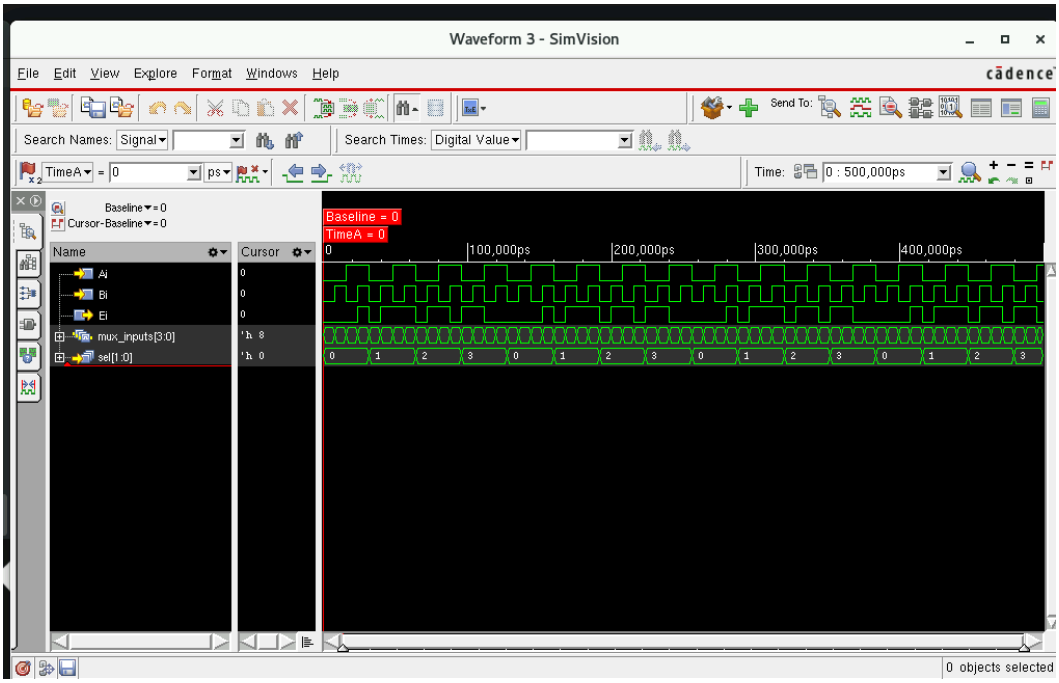



crimson:/eecs/home/aali02/EECS4612/termproject/Simulation
File Edit View Search Terminal Help
1 1 0 1 1 | 0010 | 0 1 | 0 1 | PASS
1 1 1 0 0 | 0010 | 1 0 | 1 0 | PASS
1 1 1 0 1 | 0010 | 0 1 | 0 1 | PASS
1 1 1 1 0 | 0010 | 1 0 | 1 0 | PASS
1 1 1 1 1 | 0010 | 0 1 | 0 1 | PASS
0 0 0 0 0 | 0011 | 1 0 | 1 0 | PASS
0 0 0 0 1 | 0011 | 0 1 | 0 1 | PASS
0 0 0 1 0 | 0011 | 1 0 | 1 0 | PASS
0 0 0 1 1 | 0011 | 0 1 | 0 1 | PASS
0 0 1 0 0 | 0011 | 1 0 | 1 0 | PASS
0 0 1 0 1 | 0011 | 0 1 | 0 1 | PASS
0 0 1 1 0 | 0011 | 1 0 | 1 0 | PASS
0 0 1 1 1 | 0011 | 0 1 | 0 1 | PASS
0 1 0 0 0 | 0011 | 1 0 | 1 0 | PASS
0 1 0 0 1 | 0011 | 0 1 | 0 1 | PASS
0 1 0 1 0 | 0011 | 1 0 | 1 0 | PASS
0 1 0 1 1 | 0011 | 0 1 | 0 1 | PASS
0 1 1 0 0 | 0011 | 1 0 | 1 0 | PASS
0 1 1 0 1 | 0011 | 0 1 | 0 1 | PASS
0 1 1 1 0 | 0011 | 1 0 | 1 0 | PASS
0 1 1 1 1 | 0011 | 0 1 | 0 1 | PASS
1 0 0 0 0 | 0011 | 0 1 | 0 1 | PASS
1 0 0 0 1 | 0011 | 1 1 | 1 1 | PASS
Design Browser
Objects
Methods

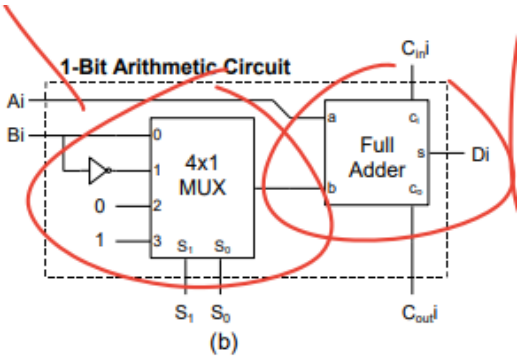

xcelium> run 500 ns
Functional Verification: logic_unit
Ai Bi | sel | Ei exp_Ei | Operation | Result
0 0 | 00 | 0 0 | AND | PASS
0 1 | 00 | 0 0 | AND | PASS
1 0 | 00 | 0 0 | AND | PASS
1 1 | 00 | 1 1 | AND | PASS
0 0 | 01 | 0 0 | OR | PASS
0 1 | 01 | 1 1 | OR | PASS
1 0 | 01 | 1 1 | OR | PASS
1 1 | 01 | 1 1 | OR | PASS
0 0 | 10 | 0 0 | XOR | PASS
0 1 | 10 | 1 1 | XOR | PASS
1 0 | 10 | 1 1 | XOR | PASS
1 1 | 10 | 0 0 | XOR | PASS
0 0 | 11 | 1 1 | NOT(Ai) | PASS
0 1 | 11 | 1 1 | NOT(Ai) | PASS
1 0 | 11 | 0 0 | NOT(Ai) | PASS
1 1 | 11 | 0 0 | NOT(Ai) | PASS
All logic_unit tests PASSED.

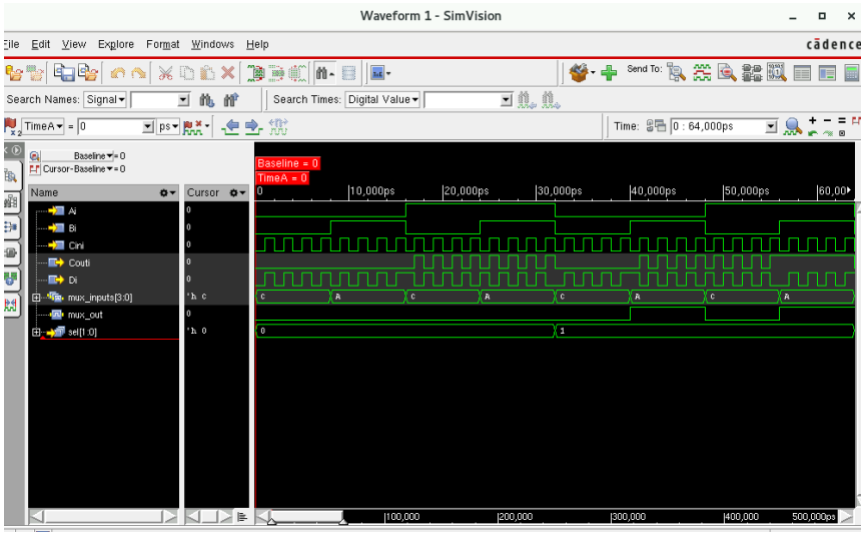
The waveform of the Logic unit is shown below. It shows the correct working according to our truth table and testbench:

Arithmetic Unit:



The arithmetic unit is designed to do arithmetic operations using the 4-1 mux we designed earlier as well as a full adder which I also designed. The architecture of the arithmetic circuit is shown through this image on the right.





In the image above, we can see the simulated waveform of the Arithmetic Unit. The truth table below shows what's happening in the circuit.

S0	S1	Cin	Operation	Function
0	0	0	F = A	Transfer A
0	0	1	F = A +1	Increment A
1	0	0	F = A +B	Add
1	0	1	F = A+B+1	Add w/ Carry
0	1	0	F = A ~B	Sub w/ borrow
0	1	1	F = A ~B +1	Subtract
1	1	0	F = A-1	Decrement A
1	1	1	F = A	Transfer A

```
xcelium> run 500 ns
============================================================
   Functional Verification: arithmetic_unit
============================================================
 Ai Bi Cin | sel | Di Cout | Exp_Di Exp_Cout | Operation          | Result
-------------------------------------------------------------------------------
  0  0  0 | 00  |  0   0  |   0       0     |                Ai + 0 + Cini | PASS
  0  0  1 | 00  |  1   0  |   1       0     |                Ai + 0 + Cini | PASS
  0  1  0 | 00  |  0   0  |   0       0     |                Ai + 0 + Cini | PASS
  0  1  1 | 00  |  1   0  |   1       0     |                Ai + 0 + Cini | PASS
  1  0  0 | 00  |  1   0  |   1       0     |                Ai + 0 + Cini | PASS
  1  0  1 | 00  |  0   1  |   0       1     |                Ai + 0 + Cini | PASS
  1  1  0 | 00  |  1   0  |   1       0     |                Ai + 0 + Cini | PASS
  1  1  1 | 00  |  0   1  |   0       1     |                Ai + 0 + Cini | PASS
  0  0  0 | 01  |  0   0  |   0       0     |               Ai + Bi + Cini | PASS
  0  0  1 | 01  |  1   0  |   1       0     |               Ai + Bi + Cini | PASS
  0  1  0 | 01  |  1   0  |   1       0     |               Ai + Bi + Cini | PASS
  0  1  1 | 01  |  0   1  |   0       1     |               Ai + Bi + Cini | PASS
  1  0  0 | 01  |  1   0  |   1       0     |               Ai + Bi + Cini | PASS
  1  0  1 | 01  |  0   1  |   0       1     |               Ai + Bi + Cini | PASS
  1  1  0 | 01  |  0   1  |   0       1     |               Ai + Bi + Cini | PASS
  1  1  1 | 01  |  1   1  |   1       1     |               Ai + Bi + Cini | PASS
  0  0  0 | 10  |  1   0  |   1       0     |              Ai + ~Bi + Cini | PASS
  0  0  1 | 10  |  0   1  |   0       1     |              Ai + ~Bi + Cini | PASS
  0  1  0 | 10  |  0   0  |   0       0     |              Ai + ~Bi + Cini | PASS
  0  1  1 | 10  |  1   0  |   1       0     |              Ai + ~Bi + Cini | PASS
  1  0  0 | 10  |  0   1  |   0       1     |              Ai + ~Bi + Cini | PASS
  1  0  1 | 10  |  1   1  |   1       1     |              Ai + ~Bi + Cini | PASS
  1  1  0 | 10  |  1   0  |   1       0     |              Ai + ~Bi + Cini | PASS
  1  1  1 | 10  |  0   1  |   0       1     |              Ai + ~Bi + Cini | PASS
  0  0  0 | 11  |  1   0  |   1       0     |                Ai + 1 + Cini | PASS
  0  0  1 | 11  |  0   1  |   0       1     |                Ai + 1 + Cini | PASS
  0  1  0 | 11  |  1   0  |   1       0     |                Ai + 1 + Cini | PASS
  0  1  1 | 11  |  0   1  |   0       1     |                Ai + 1 + Cini | PASS
  1  0  0 | 11  |  0   1  |   0       1     |                Ai + 1 + Cini | PASS
  1  0  1 | 11  |  1   1  |   1       1     |                Ai + 1 + Cini | PASS
  1  1  0 | 11  |  0   1  |   0       1     |                Ai + 1 + Cini | PASS
  1  1  1 | 11  |  1   1  |   1       1     |                Ai + 1 + Cini | PASS
-------------------------------------------------------------------------------
All arithmetic_unit tests PASSED.
```

Now that all the units have been individually simulated and verified, lets move on to the resulting area and power.

Area and Power of 1-bit ALU:

Power Report:

Open genus_alu_1bit_power.rep
~/EECS4612/termproject/Synthesis/reports

```
Instance: /alu_1bit
Power Unit: W
PDB Frames: /stim#0/frame#0
  ------------------------------------------------------------------------
    Category         Leakage     Internal    Switching        Total    Row%
  ------------------------------------------------------------------------
       memory     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
     register     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
        latch     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
        logic     2.11523e-09  1.68016e-08  1.18341e-06  1.20232e-06 100.00%
         bbox     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
        clock     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
          pad     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
           pm     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
  ------------------------------------------------------------------------
    Subtotal     2.11523e-09  1.68016e-08  1.18341e-06  1.20232e-06 100.00%
  Percentage           0.18%        1.40%       98.43%      100.00% 100.00%
  ------------------------------------------------------------------------
```

This is the power report of the ALU. We can see the total power consumed is 1.202 x 10^-6 W and switching takes up around 98% of the power. Very low leakage power as well showing efficiency.

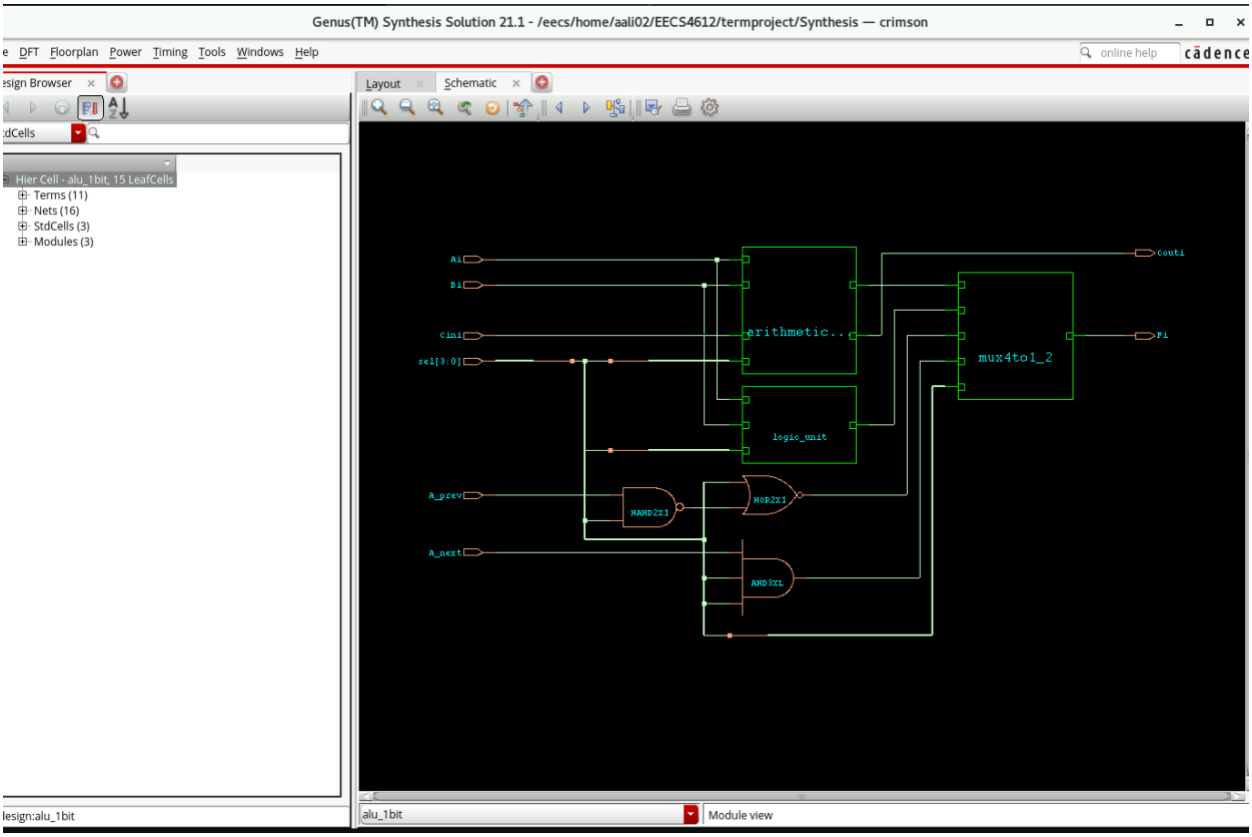
Area Report:

Open genus_alu_1bit_area.rep
~/EECS4612/termproject/Synthesis/reports

```
====================================================
  Generated by:           Genus(TM) Synthesis Solution 21.17-s066_1
  Generated on:           Dec 12 2025  07:25:02 pm
  Module:                 alu_1bit
  Operating conditions:   PVT_0P9V_125C (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================


   Gate    Instances   Area      Library
-----------------------------------------
ADDFHXL            1   7.524    slow_vdd1v0
AND2XL             1   1.368    slow_vdd1v0
AND3XL             1   2.052    slow_vdd1v0
CLKBUFX20          2  16.416    slow_vdd1v0
CLKMX2X12          1   6.498    slow_vdd1v0
INVXL              1   0.684    slow_vdd1v0
MX2X1              1   2.394    slow_vdd1v0
MX2XL              2   4.788    slow_vdd1v0
MX4XL              1   7.182    slow_vdd1v0
NAND2X1            1   1.026    slow_vdd1v0
NOR2BX1            1   1.368    slow_vdd1v0
NOR2X1             1   1.026    slow_vdd1v0
OR2XL              1   1.368    slow_vdd1v0
-----------------------------------------
total             15  53.694


     Type      Instances  Area  Area %
---------------------------------------
inverter               1  0.684    1.3
buffer                 2 16.416   30.6
logic                 12 36.594   68.2
physical_cells         0  0.000    0.0
---------------------------------------
total                 15 53.694  100.0
```

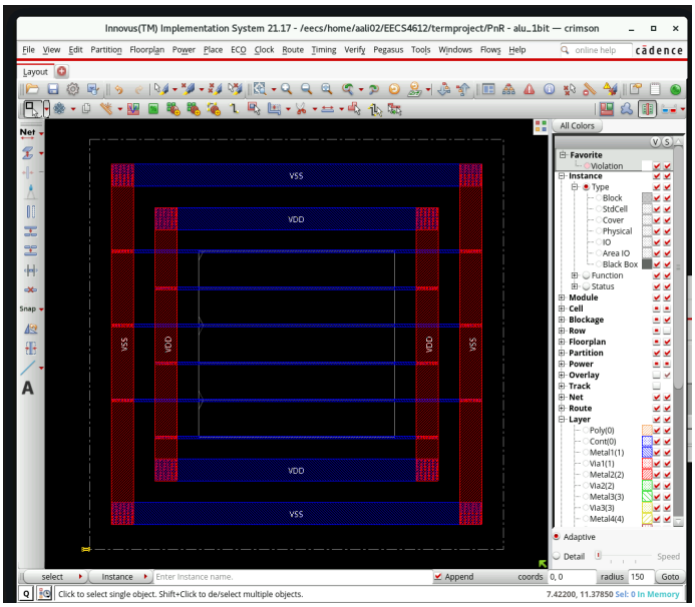
The total area of the 1-bit ALU is 53.694 x 10^-6 meters squared. Other details pertaining to the area are also shown in the diagram.

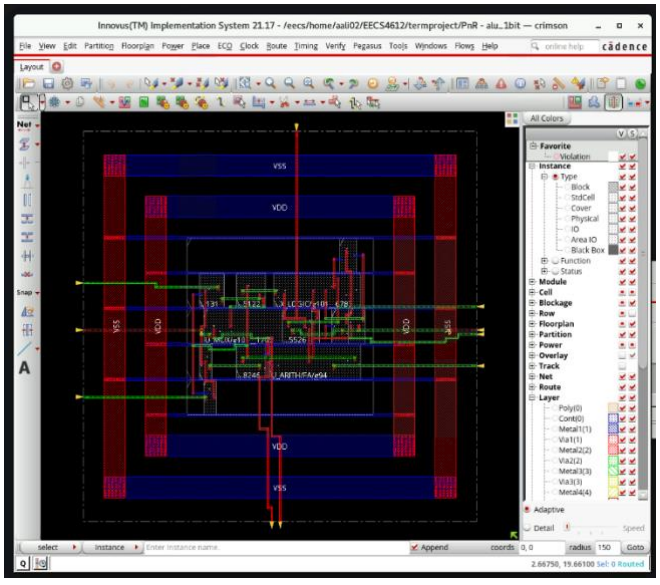
Gate Level Netlist Schematic:



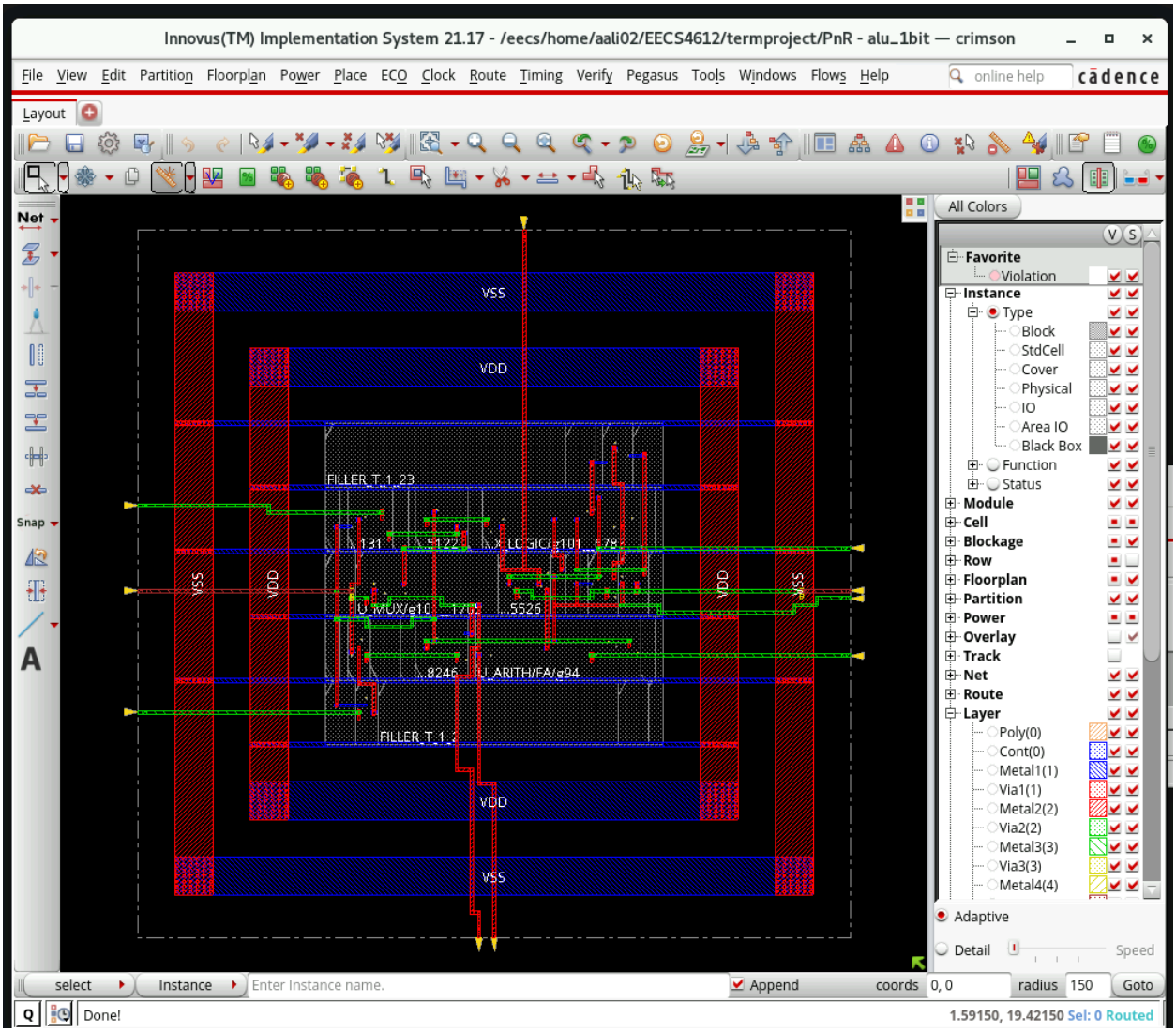
The figure above shows the schematic of the synthesized 1-bit ALU. It has all the functional units that we created before.

Physical Design of 1-bit ALU

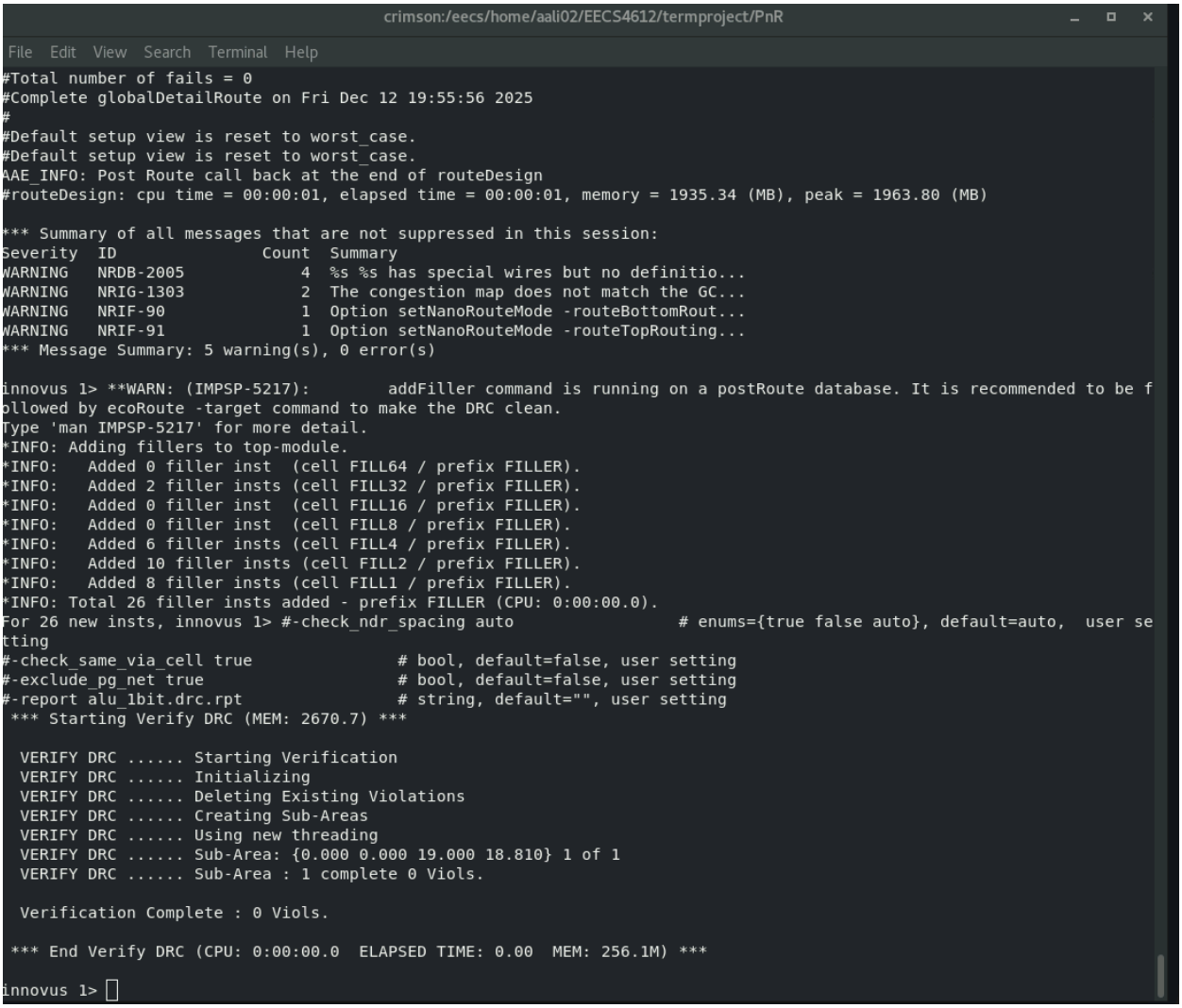


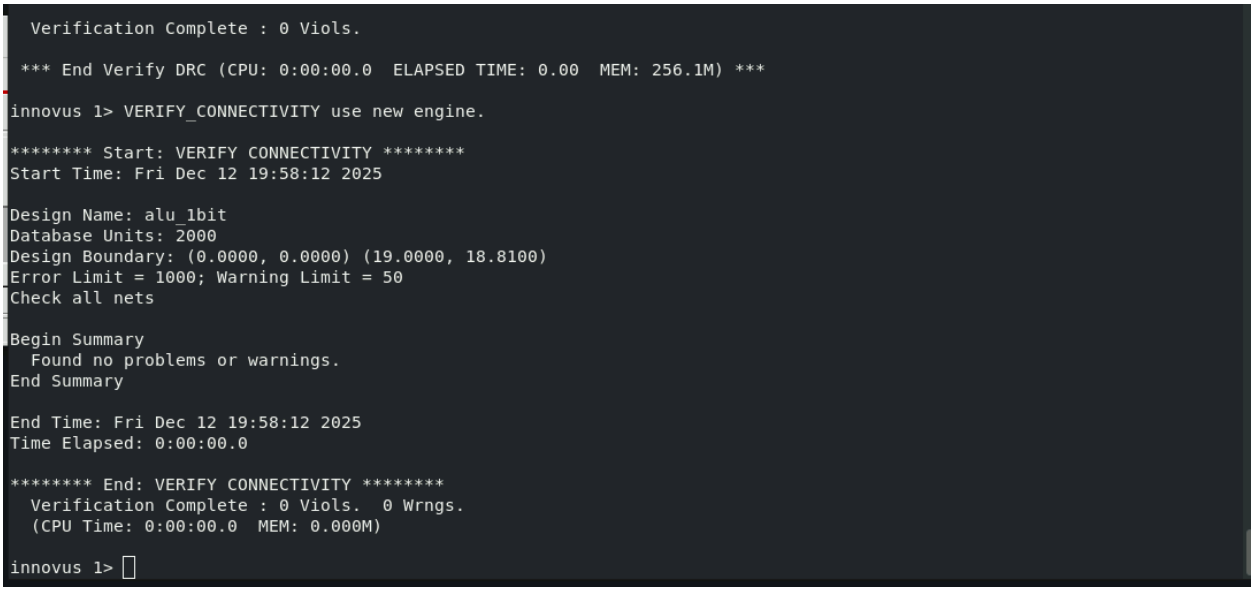


The above screenshots show the power routing and standard cell placement. The screenshot below shows the final routing with filler.



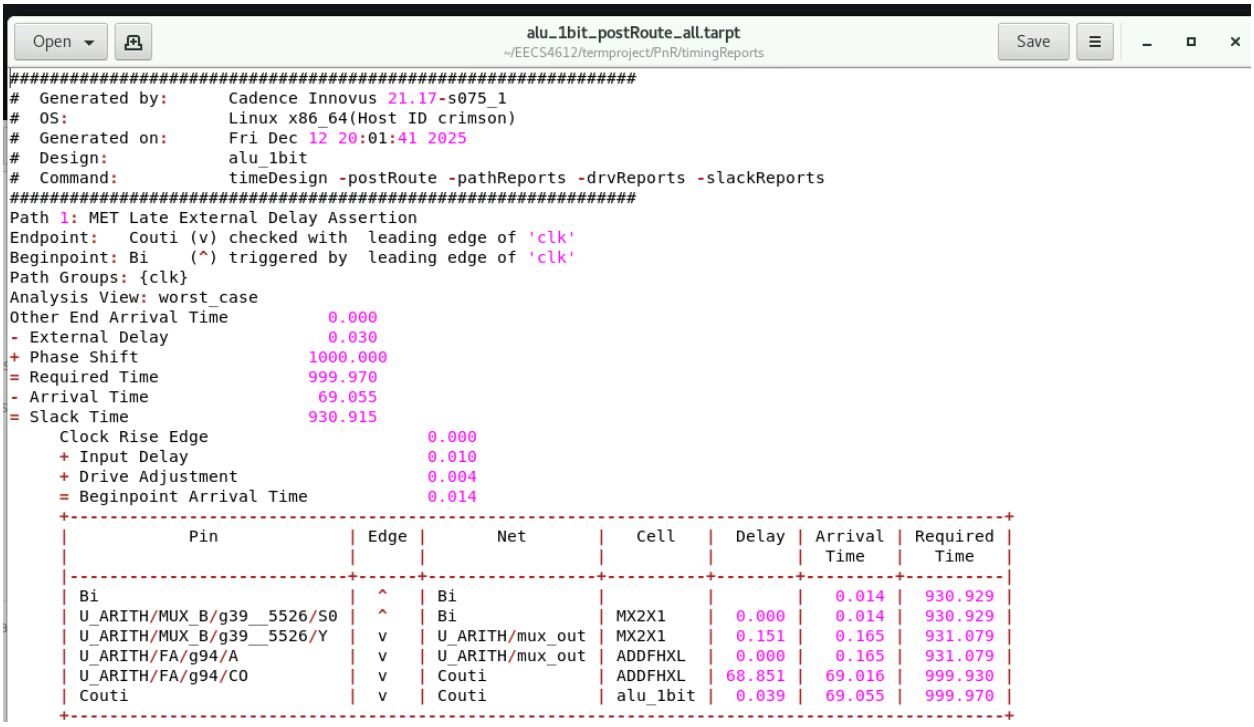
DRC and Connectivity Verification


crimson:/eecs/home/aali02/EECS4612/termproject/PnR
File Edit View Search Terminal Help
#Total number of fails = 0
#Complete globalDetailRoute on Fri Dec 12 19:55:56 2025
#
#Default setup view is reset to worst_case.
#Default setup view is reset to worst_case.
AAE_INFO: Post Route call back at the end of routeDesign
#routeDesign: cpu time = 00:00:01, elapsed time = 00:00:01, memory = 1935.34 (MB), peak = 1963.80 (MB)
*** Summary of all messages that are not suppressed in this session:
Severity ID Count Summary
WARNING NRDB-2005 4 %s %s has special wires but no definitio...
WARNING NRIG-1303 2 The congestion map does not match the GC...
WARNING NRIF-90 1 Option setNanoRouteMode -routeBottomRout...
WARNING NRIF-91 1 Option setNanoRouteMode -routeTopRouting...
*** Message Summary: 5 warning(s), 0 error(s)
innovus 1> **WARN: (IMPSP-5217): addFiller command is running on a postRoute database. It is recommended to be f
ollowed by ecoRoute -target command to make the DRC clean.
Type 'man IMPSP-5217' for more detail.
*INFO: Adding fillers to top-module.
*INFO: Added 0 filler inst (cell FILL64 / prefix FILLER).
*INFO: Added 2 filler insts (cell FILL32 / prefix FILLER).
*INFO: Added 0 filler inst (cell FILL16 / prefix FILLER).
*INFO: Added 0 filler inst (cell FILL8 / prefix FILLER).
*INFO: Added 6 filler insts (cell FILL4 / prefix FILLER).
*INFO: Added 10 filler insts (cell FILL2 / prefix FILLER).
*INFO: Added 8 filler insts (cell FILL1 / prefix FILLER).
*INFO: Total 26 filler insts added - prefix FILLER (CPU: 0:00:00.0).
For 26 new insts, innovus 1> #-check_ndr_spacing auto # enums={true false auto}, default=auto, user se
tting
#-check_same_via_cell true # bool, default=false, user setting
#-exclude_pg_net true # bool, default=false, user setting
#-report alu_1bit.drc.rpt # string, default="", user setting
*** Starting Verify DRC (MEM: 2670.7) ***
VERIFY DRC ..... Starting Verification
VERIFY DRC ..... Initializing
VERIFY DRC ..... Deleting Existing Violations
VERIFY DRC ..... Creating Sub-Areas
VERIFY DRC ..... Using new threading
VERIFY DRC ..... Sub-Area: {0.000 0.000 19.000 18.810} 1 of 1
VERIFY DRC ..... Sub-Area : 1 complete 0 Viols.
Verification Complete : 0 Viols.
*** End Verify DRC (CPU: 0:00:00.0 ELAPSED TIME: 0.00 MEM: 256.1M) ***
innovus 1>


Verification Complete : 0 Viols.
*** End Verify DRC (CPU: 0:00:00.0 ELAPSED TIME: 0.00 MEM: 256.1M) ***
innovus 1> VERIFY_CONNECTIVITY use new engine.
******** Start: VERIFY CONNECTIVITY ********
Start Time: Fri Dec 12 19:58:12 2025
Design Name: alu_1bit
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (19.0000, 18.8100)
Error Limit = 1000; Warning Limit = 50
Check all nets
Begin Summary
Found no problems or warnings.
End Summary
End Time: Fri Dec 12 19:58:12 2025
Time Elapsed: 0:00:00.0
******** End: VERIFY CONNECTIVITY ********
Verification Complete : 0 Viols. 0 Wrngs.
(CPU Time: 0:00:00.0 MEM: 0.000M)
innovus 1>

Estimating Speed using Timing Analysis:

Since we cannot do clock tree synthesis because there is no clock, I used timing analysis.



alu_1bit_postRoute_all.tarpt
~/EECS4612/termproject/PnR/timingReports

```
#############################################################
#  Generated by:      Cadence Innovus 21.17-s075_1
#  OS:                Linux x86_64(Host ID crimson)
#  Generated on:      Fri Dec 12 20:01:41 2025
#  Design:            alu_1bit
#  Command:           timeDesign -postRoute -pathReports -drvReports -slackReports
#############################################################
Path 1: MET Late External Delay Assertion
Endpoint:   Couti (v) checked with  leading edge of 'clk'
Beginpoint: Bi    (^) triggered by  leading edge of 'clk'
Path Groups: {clk}
Analysis View: worst_case
Other End Arrival Time          0.000
- External Delay                0.030
+ Phase Shift                1000.000
= Required Time              999.970
- Arrival Time                69.055
= Slack Time                 930.915
     Clock Rise Edge                      0.000
     + Input Delay                        0.010
     + Drive Adjustment                   0.004
     = Beginpoint Arrival Time            0.014
     +-------------------------------------------------------------------------------------------+
     |            Pin             | Edge |       Net        |   Cell   |  Delay | Arrival | Required |
     |                            |      |                  |          |        |  Time   |   Time   |
     |----------------------------+------+------------------+----------+--------+---------+----------|
     | Bi                         |  ^   | Bi               |          |        |   0.014 |  930.929 |
     | U_ARITH/MUX_B/g39__5526/S0 |  ^   | Bi               | MX2X1    |  0.000 |   0.014 |  930.929 |
     | U_ARITH/MUX_B/g39__5526/Y  |  v   | U_ARITH/mux_out  | MX2X1    |  0.151 |   0.165 |  931.079 |
     | U_ARITH/FA/g94/A           |  v   | U_ARITH/mux_out  | ADDFHXL  |  0.000 |   0.165 |  931.079 |
     | U_ARITH/FA/g94/CO          |  v   | Couti            | ADDFHXL  | 68.851 |  69.016 |  999.930 |
     | Couti                      |  v   | Couti            | alu_1bit |  0.039 |  69.055 |  999.970 |
     +-------------------------------------------------------------------------------------------+
```

Estimate Max Speed: we have the arrival time so we need to simply do $fmax = 1$ / critical path delay

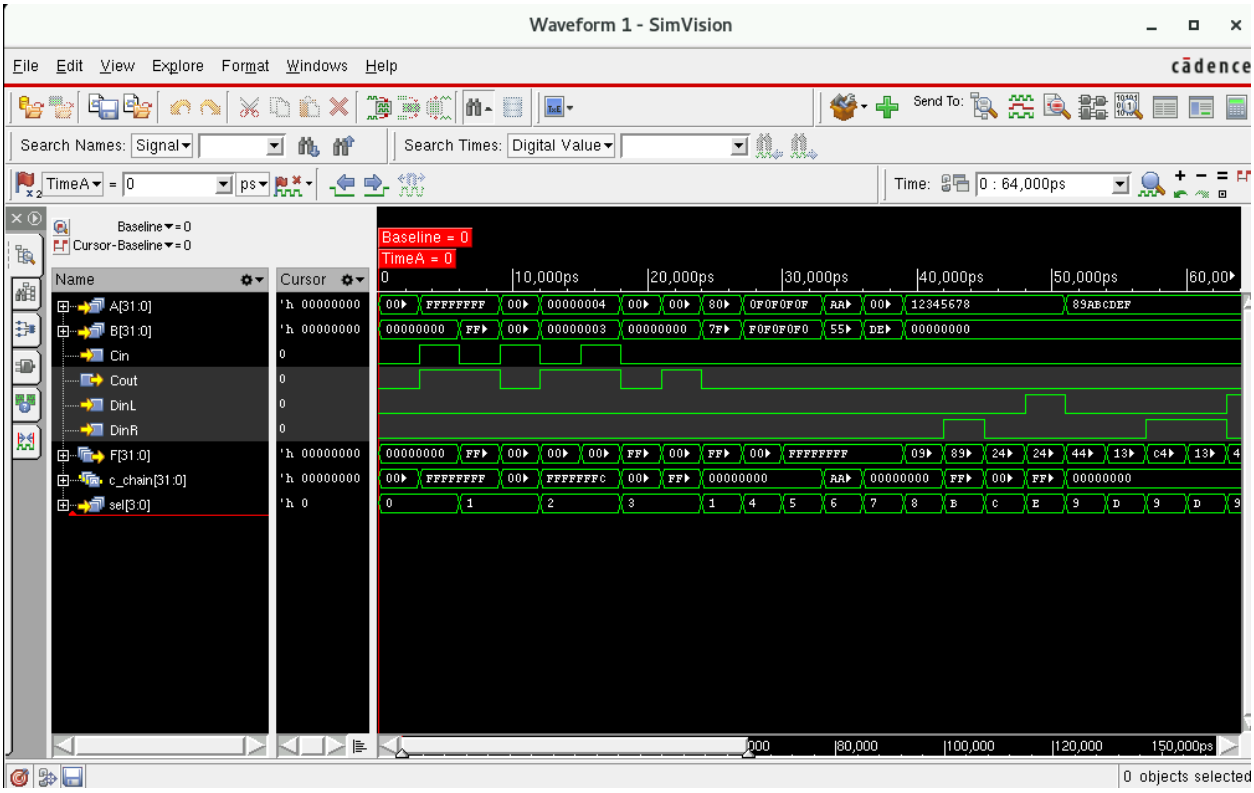
$$fmax = \frac{1}{69.055\ x\ 10^{-9}} = 14.48\ mhz$$

So, the total max speed the circuit can operate under is 14.48 mhz.

Part 2: 32-Bit ALUS

Modular 32-Bit ALU:

The format is as follows: information about the modular ALU followed by information about the behavioral ALU.



The above screenshot shows the waveform for the modular ALU (1-bit instantiated 32 times). We can see that everything works in the waveform. Below is a couple screenshots of the testbench verification.

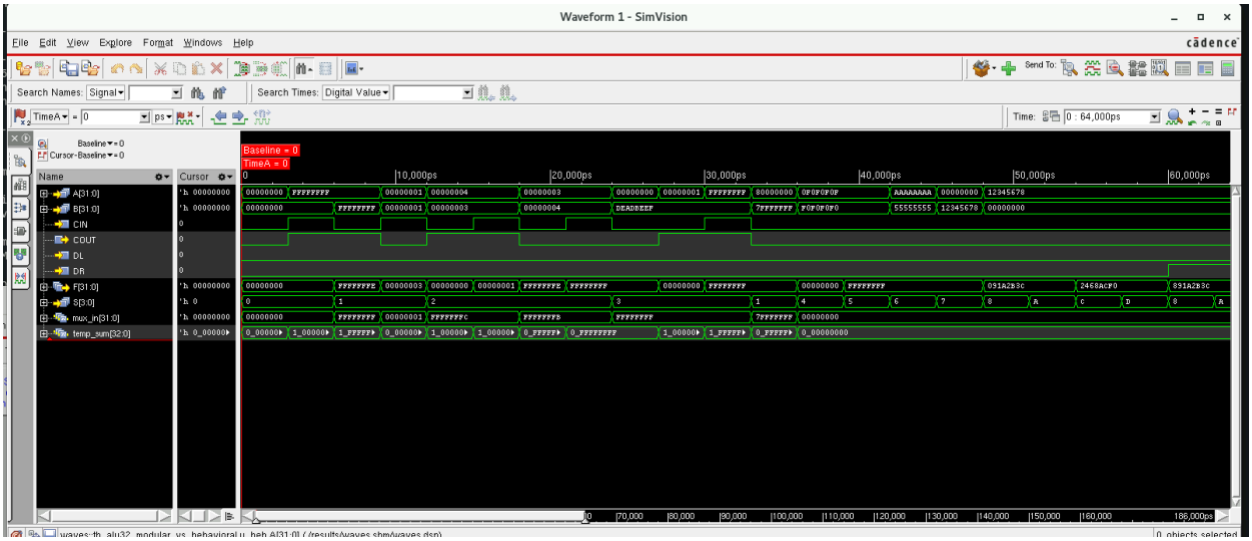
```
============================================================
   Functional Verification: 32-bit Modular ALU (self-check)
============================================================

---- Arithmetic tests ----
------------------------------------------------------------
ARITH: Transfer A | sel=0x0  A=0x00000000  B=0x00000000  Cin=0 DinL=0 DinR=0
Expected: F=0x00000000 Cout=0 | Got: F=0x00000000 Cout=0
[PASS] Arithmetic
------------------------------------------------------------

------------------------------------------------------------
ARITH: Increment A (Cin=1) | sel=0x0  A=0xffffffff  B=0x00000000  Cin=1 DinL=0 DinR=0
Expected: F=0x00000000 Cout=1 | Got: F=0x00000000 Cout=1
[PASS] Arithmetic
------------------------------------------------------------

------------------------------------------------------------
ARITH: A + B (max) | sel=0x1  A=0xffffffff  B=0xffffffff  Cin=0 DinL=0 DinR=0
Expected: F=0xfffffffe Cout=1 | Got: F=0xfffffffe Cout=1
[PASS] Arithmetic
------------------------------------------------------------

------------------------------------------------------------
ARITH: A + B + 1 | sel=0x1  A=0x00000001  B=0x00000001  Cin=1 DinL=0 DinR=0
Expected: F=0x00000003 Cout=0 | Got: F=0x00000003 Cout=0
[PASS] Arithmetic
------------------------------------------------------------

------------------------------------------------------------
ARITH: A - B - 1 (Cin=0) | sel=0x2  A=0x00000004  B=0x00000003  Cin=0 DinL=0 DinR=0
Expected: F=0x00000000 Cout=1 | Got: F=0x00000000 Cout=1
[PASS] Arithmetic
------------------------------------------------------------

------------------------------------------------------------
ARITH: A - B (Cin=1) | sel=0x2  A=0x00000004  B=0x00000003  Cin=1 DinL=0 DinR=0
Expected: F=0x00000001 Cout=1 | Got: F=0x00000001 Cout=1
[PASS] Arithmetic
```

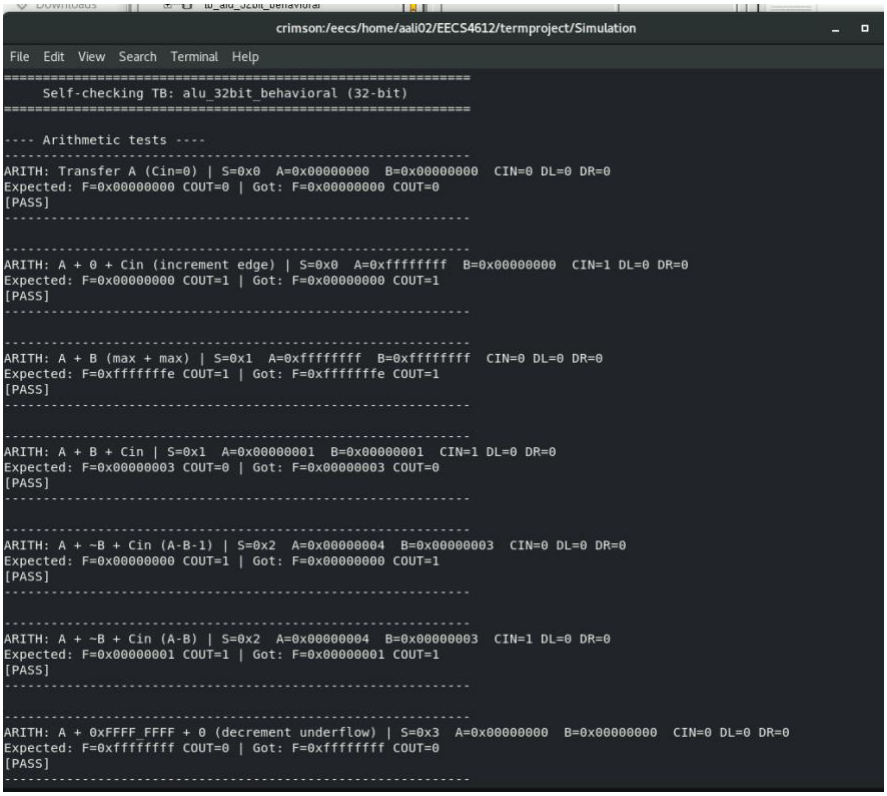
```
File  Edit  View  Search  Terminal  Help
------------------------------------------------------------

------------------------------------------------------------
LOGIC: NOT A | sel=0x7  A=0x00000000  B=0xdeadbeef  Cin=0 DinL=0 DinR=0
Expected: F=0xffffffff Cout=0 | Got: F=0xffffffff Cout=0
[PASS] Logic/Shift
------------------------------------------------------------

---- Shift tests ----
------------------------------------------------------------
SHIFT: Right (DR=0) | sel=0x8  A=0x12345678  B=0x00000000  Cin=0 DinL=0 DinR=0
Expected: F=0x091a2b3c Cout=0 | Got: F=0x091a2b3c Cout=0
[PASS] Logic/Shift
------------------------------------------------------------

------------------------------------------------------------
SHIFT: Right (DR=1, sel varied) | sel=0xb  A=0x12345678  B=0x00000000  Cin=0 DinL=0 DinR=1
Expected: F=0x891a2b3c Cout=0 | Got: F=0x891a2b3c Cout=0
[PASS] Logic/Shift
------------------------------------------------------------

------------------------------------------------------------
SHIFT: Left (DL=0) | sel=0xc  A=0x12345678  B=0x00000000  Cin=0 DinL=0 DinR=0
Expected: F=0x2468acf0 Cout=0 | Got: F=0x2468acf0 Cout=0
[PASS] Logic/Shift
------------------------------------------------------------

------------------------------------------------------------
SHIFT: Left (DL=1, sel varied) | sel=0xe  A=0x12345678  B=0x00000000  Cin=0 DinL=1 DinR=0
Expected: F=0x2468acf1 Cout=0 | Got: F=0x2468acf1 Cout=0
[PASS] Logic/Shift
------------------------------------------------------------

------------------------------------------------------------
SHIFT SWEEP: Right | sel=0x9  A=0x89abcdef  B=0x00000000  Cin=0 DinL=0 DinR=0
Expected: F=0x44d5e6f7 Cout=0 | Got: F=0x44d5e6f7 Cout=0
[PASS] Logic/Shift
------------------------------------------------------------

------------------------------------------------------------
```

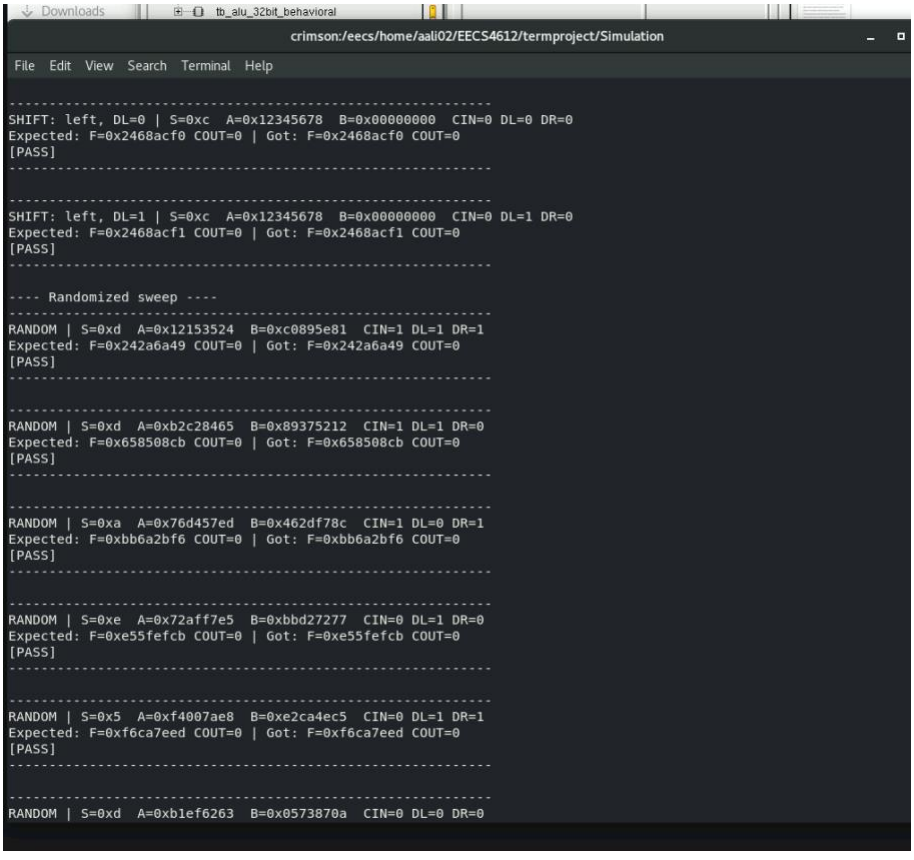
```
crimson:/eecs/home/aali02/EECS4612/termproject/Simulation
File  Edit  View  Search  Terminal  Help
------------------------------------------------------------


------------------------------------------------------------
RANDOM | sel=0xa  A=0xafd8565f  B=0x22290d44  Cin=1 DinL=1 DinR=0
Expected: F=0x57ec2b2f Cout=0 | Got: F=0x57ec2b2f Cout=0
[PASS] Logic/Shift
------------------------------------------------------------

------------------------------------------------------------
RANDOM | sel=0xf  A=0x14cfc129  B=0xf682e2ed  Cin=0 DinL=1 DinR=1
Expected: F=0x299f8253 Cout=0 | Got: F=0x299f8253 Cout=0
[PASS] Logic/Shift
------------------------------------------------------------

------------------------------------------------------------
RANDOM | sel=0xe  A=0x3cf11979  B=0x2231ff44  Cin=0 DinL=0 DinR=1
Expected: F=0x79e232f2 Cout=0 | Got: F=0x79e232f2 Cout=0
[PASS] Logic/Shift
------------------------------------------------------------

------------------------------------------------------------
RANDOM | sel=0xe  A=0x6e5daddc  B=0xcd5ebc9a  Cin=1 DinL=1 DinR=0
Expected: F=0xdcbb5bb9 Cout=0 | Got: F=0xdcbb5bb9 Cout=0
[PASS] Logic/Shift
------------------------------------------------------------

------------------------------------------------------------
RANDOM | sel=0x8  A=0xb3d97667  B=0x8531340a  Cin=0 DinL=0 DinR=1
Expected: F=0xd9ecbb33 Cout=0 | Got: F=0xd9ecbb33 Cout=0
[PASS] Logic/Shift
------------------------------------------------------------

------------------------------------------------------------
RANDOM | sel=0xd  A=0x4a74bf94  B=0x49c65d93  Cin=0 DinL=1 DinR=1
Expected: F=0x94e97f29 Cout=0 | Got: F=0x94e97f29 Cout=0
[PASS] Logic/Shift
------------------------------------------------------------


All functional verification tests completed.
```

There are quite a few tests, so I did not have space to put them all, but they pass. You can also run the testbench for yourself to verify.

Behavioral 32-bit ALU:



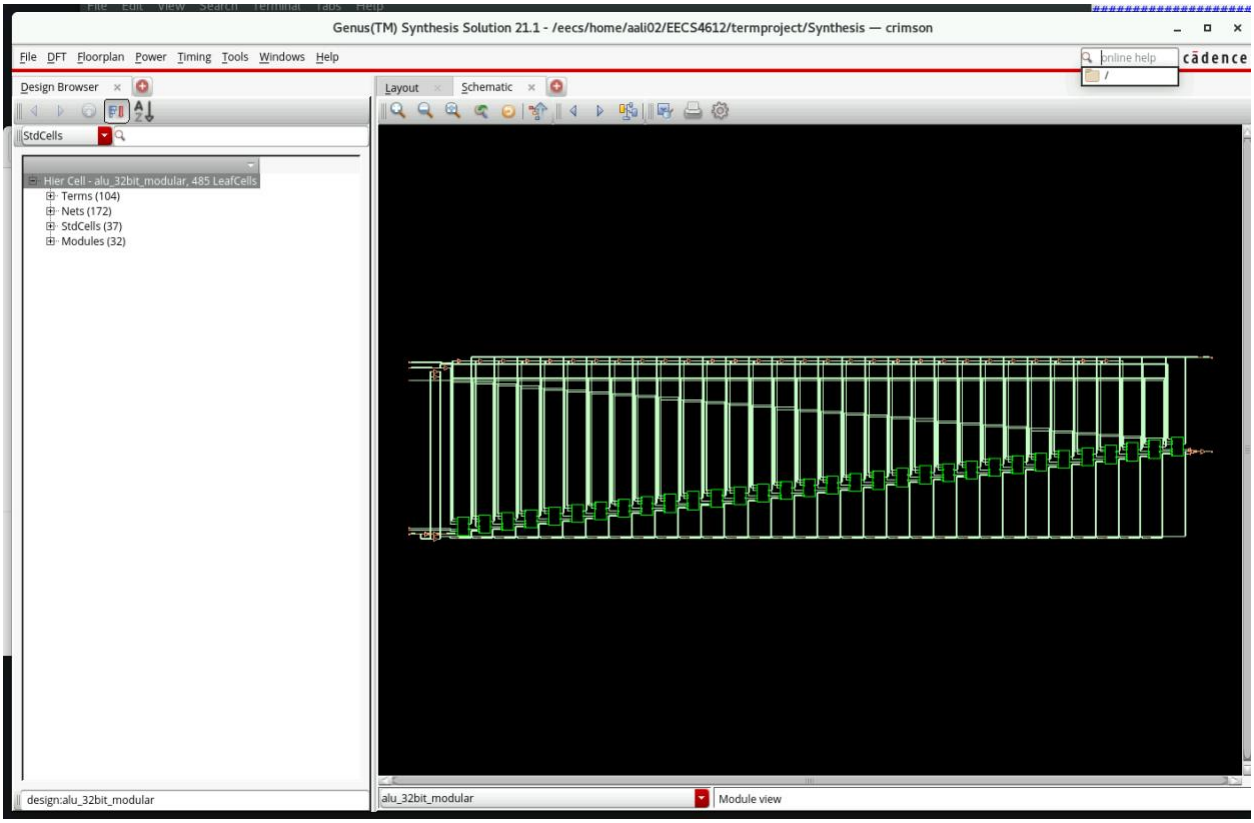

crimson:/eecs/home/aali02/EECS4612/termproject/Simulation
File Edit View Search Terminal Help
Self-checking TB: alu_32bit_behavioral (32-bit)
---- Arithmetic tests ----
ARITH: Transfer A (Cin=0) | S=0x0 A=0x00000000 B=0x00000000 CIN=0 DL=0 DR=0
Expected: F=0x00000000 COUT=0 | Got: F=0x00000000 COUT=0
[PASS]
ARITH: A + 0 + Cin (increment edge) | S=0x0 A=0xffffffff B=0x00000000 CIN=1 DL=0 DR=0
Expected: F=0x00000000 COUT=1 | Got: F=0x00000000 COUT=1
[PASS]
ARITH: A + B (max + max) | S=0x1 A=0xffffffff B=0xffffffff CIN=0 DL=0 DR=0
Expected: F=0xfffffffe COUT=1 | Got: F=0xfffffffe COUT=1
[PASS]
ARITH: A + B + Cin | S=0x1 A=0x00000001 B=0x00000001 CIN=1 DL=0 DR=0
Expected: F=0x00000003 COUT=0 | Got: F=0x00000003 COUT=0
[PASS]
ARITH: A + ~B + Cin (A-B-1) | S=0x2 A=0x00000004 B=0x00000003 CIN=0 DL=0 DR=0
Expected: F=0x00000000 COUT=1 | Got: F=0x00000000 COUT=1
[PASS]
ARITH: A + ~B + Cin (A-B) | S=0x2 A=0x00000004 B=0x00000003 CIN=1 DL=0 DR=0
Expected: F=0x00000001 COUT=1 | Got: F=0x00000001 COUT=1
[PASS]
ARITH: A + 0xFFFF_FFFF + 0 (decrement underflow) | S=0x3 A=0x00000000 B=0x00000000 CIN=0 DL=0 DR=0
Expected: F=0xffffffff COUT=0 | Got: F=0xffffffff COUT=0
[PASS]

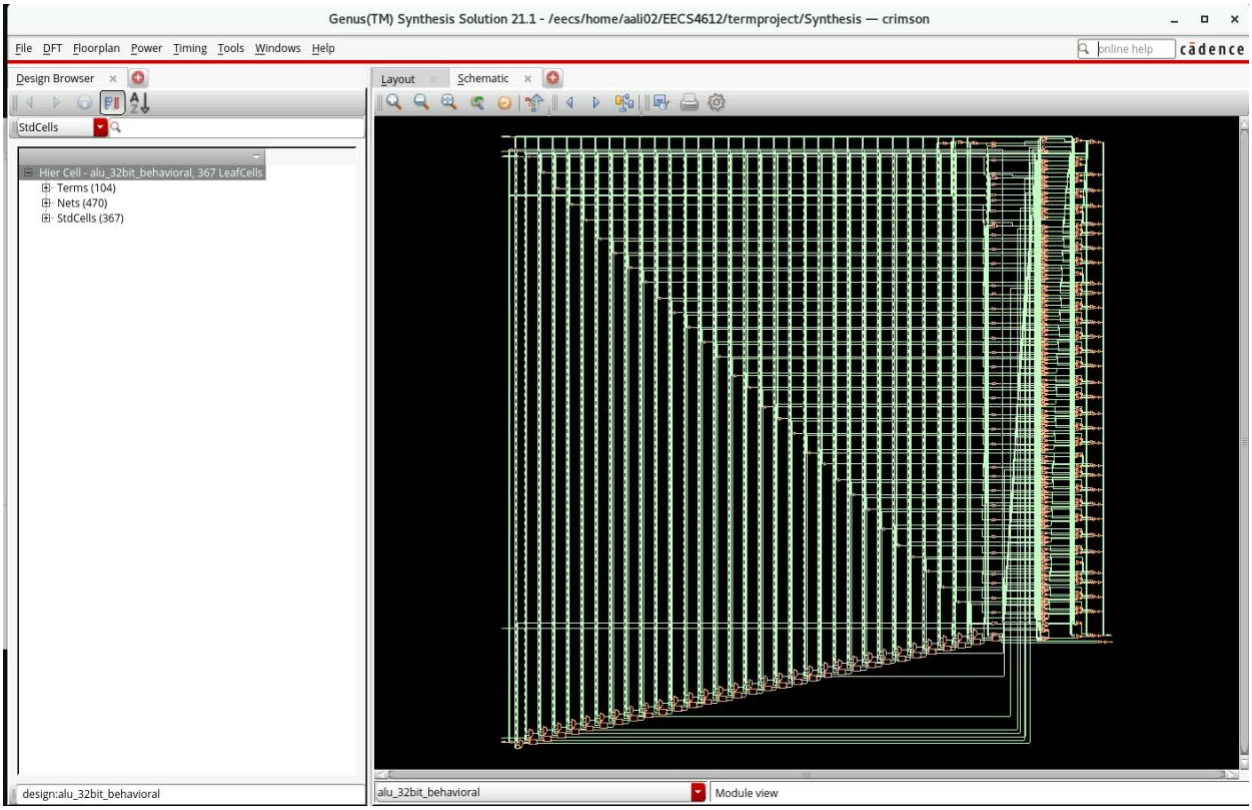

Downloads
tb_alu_32bit_behavioral
crimson:/eecs/home/aali02/EECS4612/termproject/Simulation
File Edit View Search Terminal Help
SHIFT: left, DL=0 | S=0xc A=0x12345678 B=0x00000000 CIN=0 DL=0 DR=0
Expected: F=0x2468acf0 COUT=0 | Got: F=0x2468acf0 COUT=0
[PASS]
SHIFT: left, DL=1 | S=0xc A=0x12345678 B=0x00000000 CIN=0 DL=1 DR=0
Expected: F=0x2468acf1 COUT=0 | Got: F=0x2468acf1 COUT=0
[PASS]
---- Randomized sweep ----
RANDOM | S=0xd A=0x12153524 B=0xc0895e81 CIN=1 DL=1 DR=1
Expected: F=0x242a6a49 COUT=0 | Got: F=0x242a6a49 COUT=0
[PASS]
RANDOM | S=0xd A=0xb2c28465 B=0x89375212 CIN=1 DL=1 DR=0
Expected: F=0x658508cb COUT=0 | Got: F=0x658508cb COUT=0
[PASS]
RANDOM | S=0xa A=0x76d457ed B=0x462df78c CIN=1 DL=0 DR=1
Expected: F=0xbb6a2bf6 COUT=0 | Got: F=0xbb6a2bf6 COUT=0
[PASS]
RANDOM | S=0xe A=0x72aff7e5 B=0xbbd27277 CIN=0 DL=1 DR=0
Expected: F=0xe55fefcb COUT=0 | Got: F=0xe55fefcb COUT=0
[PASS]
RANDOM | S=0x5 A=0xf4007ae8 B=0xe2ca4ec5 CIN=0 DL=1 DR=1
Expected: F=0xf6ca7eed COUT=0 | Got: F=0xf6ca7eed COUT=0
[PASS]
RANDOM | S=0xd A=0xb1ef6263 B=0x0573870a CIN=0 DL=0 DR=0

```
[PASS]
------------------------------------------------------------


------------------------------------------------------------
RANDOM | S=0x9  A=0x0fd28f1f  B=0xe9ebf6d3  CIN=1 DL=0 DR=1
Expected: F=0x87e9478f COUT=0 | Got: F=0x87e9478f COUT=0
[PASS]
------------------------------------------------------------


------------------------------------------------------------
RANDOM | S=0xc  A=0x9ff2ae3f  B=0x150caf2a  CIN=0 DL=0 DR=0
Expected: F=0x3fe55c7e COUT=0 | Got: F=0x3fe55c7e COUT=0
[PASS]
------------------------------------------------------------


------------------------------------------------------------
RANDOM | S=0x3  A=0x7d3599fa  B=0x937dbc26  CIN=1 DL=1 DR=1
Expected: F=0x7d3599fa COUT=1 | Got: F=0x7d3599fa COUT=1
[PASS]
------------------------------------------------------------


------------------------------------------------------------
RANDOM | S=0xa  A=0xafd8565f  B=0x22290d44  CIN=1 DL=1 DR=0
Expected: F=0x57ec2b2f COUT=0 | Got: F=0x57ec2b2f COUT=0
[PASS]
------------------------------------------------------------


All checks complete.
```

In the above screenshots we can see the simulation and verification of the 32 Bit Behavioral ALU. This is the one that we are going to use because of the improvements in area and power which we will analyze later in the report. The 32 bit behavioral ALU is a lot better in all metrics compared to the 32 bit modular ALU.

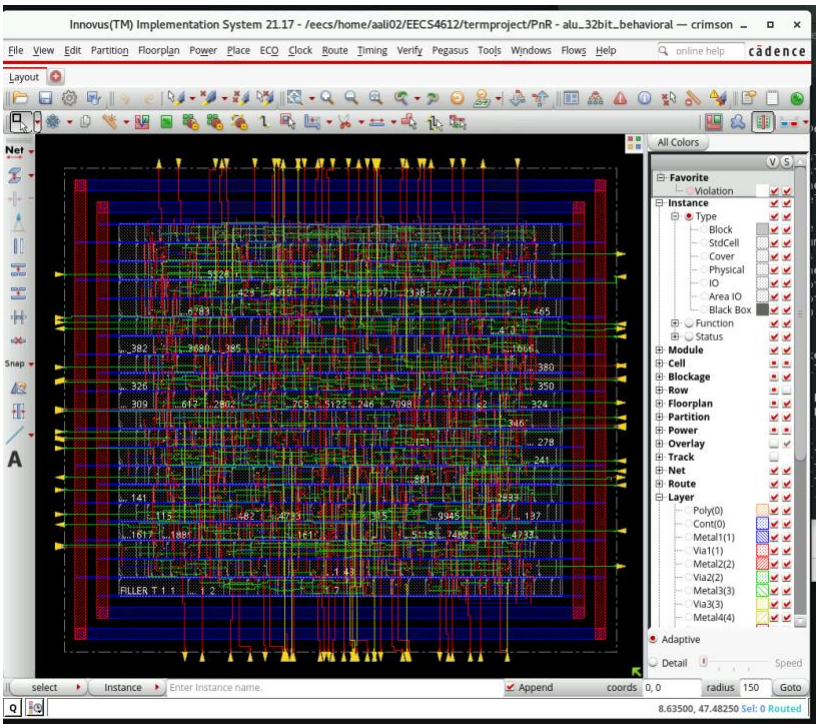
Gate level schematic of Modular and Behavioral 32-bit ALU:



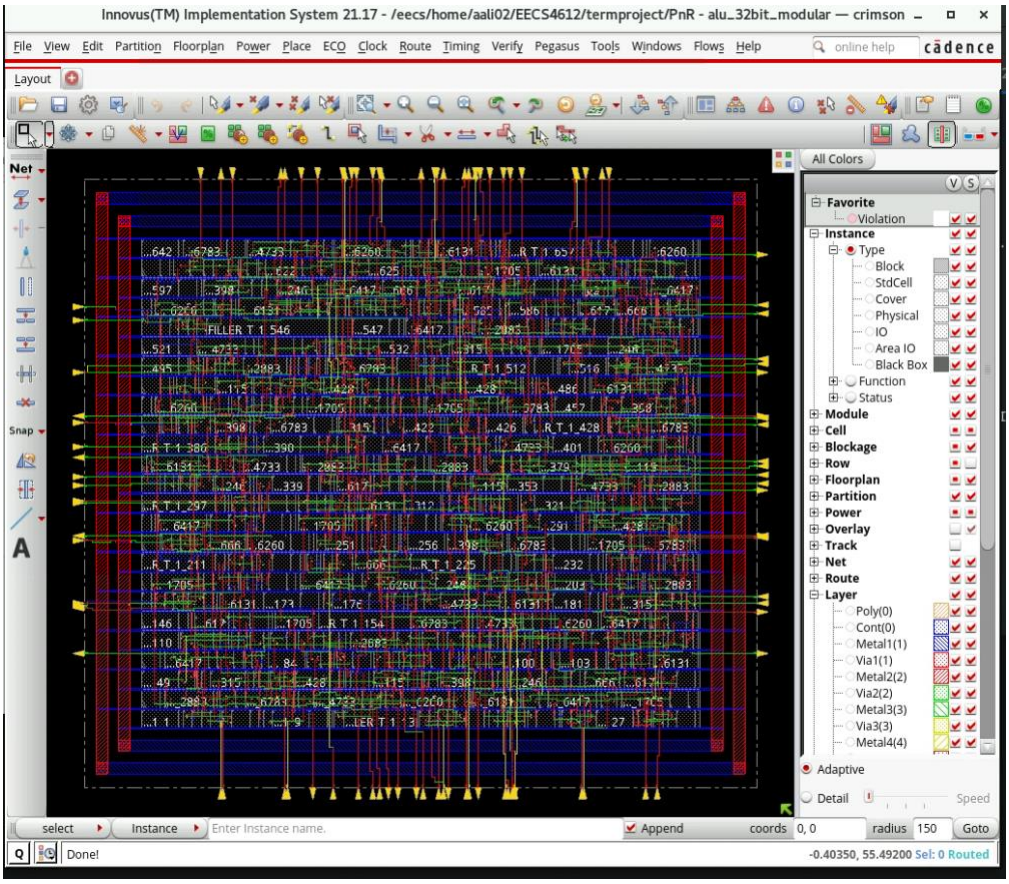


At this stage, both designs have been verified and we have checked to make sure that each component is working properly. We can now move forward

PnR Using Innovus for Modular and Behavioral Design:



Behavioral ALU Final Layout



modular ALU
Final Layout

Above we can see both screenshots of the final PnR layout with power routing, and all filler placements. This is now ready for Virtuoso. But before we do that, we need to compare both designs and pick which one will be our final. We need to do some comparison of the power, speed and area. We see this below.

Open

genus_alu_32bit_modular_power.rep
~/EECS4612/termproject/Synthesis/reports

genus_script.tcl

```
Instance: /alu_32bit_modular
Power Unit: W
PDB Frames: /stim#0/frame#0
  ------------------------------------------------------------------------------
    Category         Leakage      Internal     Switching        Total     Row%
  ------------------------------------------------------------------------------
      memory     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00    0.00%
    register     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00    0.00%
       latch     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00    0.00%
       logic     4.96023e-08  4.39078e-07  2.51057e-05  2.55943e-05  100.00%
        bbox     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00    0.00%
       clock     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00    0.00%
         pad     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00    0.00%
          pm     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00    0.00%
  ------------------------------------------------------------------------------
    Subtotal     4.96023e-08  4.39078e-07  2.51057e-05  2.55943e-05  100.00%
  Percentage           0.19%        1.72%       98.09%      100.00%  100.00%
  ------------------------------------------------------------------------------
```

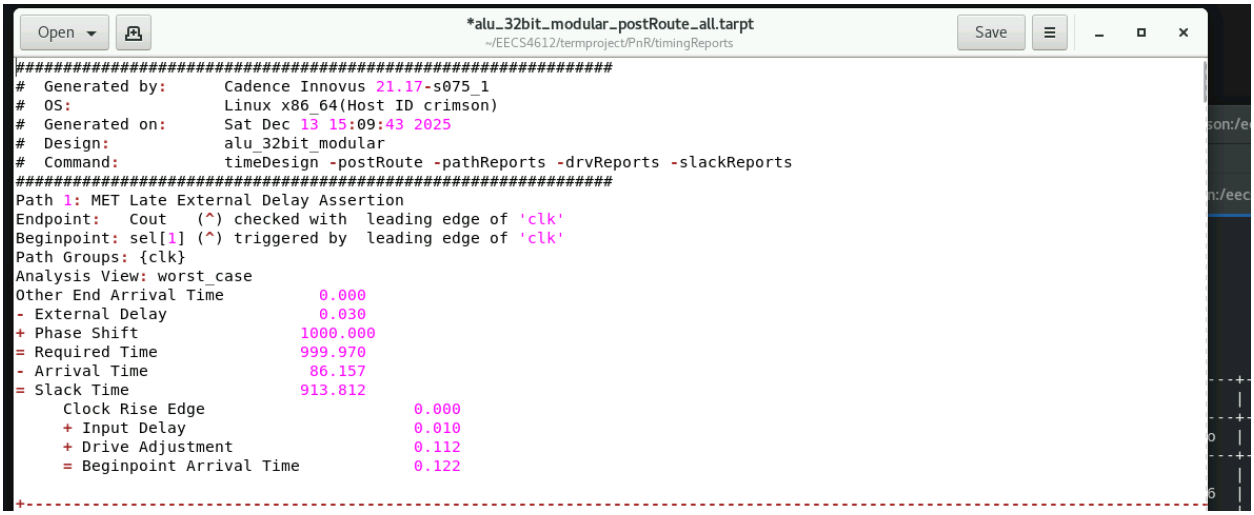
Open

genus_alu_3
~/EECS4612/t

genus_script.tcl

genus_alu_32bit_modu

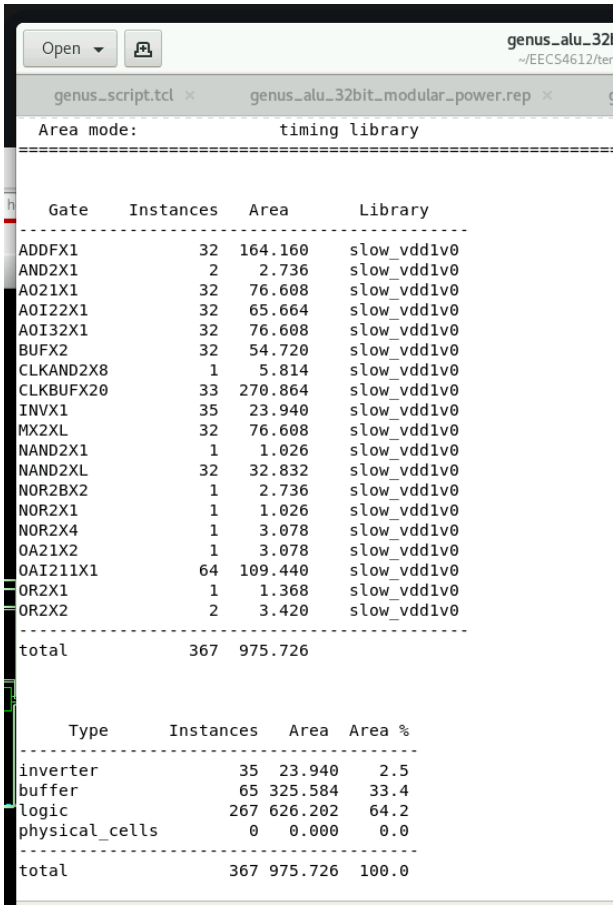
```
  Wireload mode:            enclosed
  Area mode:                timing library
==========================================================================


   Gate    Instances    Area       Library
------------------------------------------------
ADDFX1            32   164.160    slow_vdd1v0
AND2XL            32    43.776    slow_vdd1v0
AND3XL            32    65.664    slow_vdd1v0
BUFX2             31    53.010    slow_vdd1v0
BUFX3              1     2.052    slow_vdd1v0
BUFX6              1     3.078    slow_vdd1v0
CLKBUFX20         33   270.864    slow_vdd1v0
CLKINVX4           1     1.710    slow_vdd1v0
CLKMX2X12         32   207.936    slow_vdd1v0
INVX1              1     0.684    slow_vdd1v0
INVXL             32    21.888    slow_vdd1v0
MX2XL             96   229.824    slow_vdd1v0
MX4XL             32   229.824    slow_vdd1v0
NAND2X1           32    32.832    slow_vdd1v0
NOR2BX1           32    43.776    slow_vdd1v0
NOR2X1            32    32.832    slow_vdd1v0
NOR3BX2            1     3.420    slow_vdd1v0
OR2XL             32    43.776    slow_vdd1v0
------------------------------------------------
total            485  1451.106


     Type      Instances   Area   Area %
-----------------------------------------
inverter              34   24.282    1.7
buffer                66  329.004   22.7
logic                385 1097.820   75.7
physical_cells         0    0.000    0.0
-----------------------------------------
total                485 1451.106  100.0
```



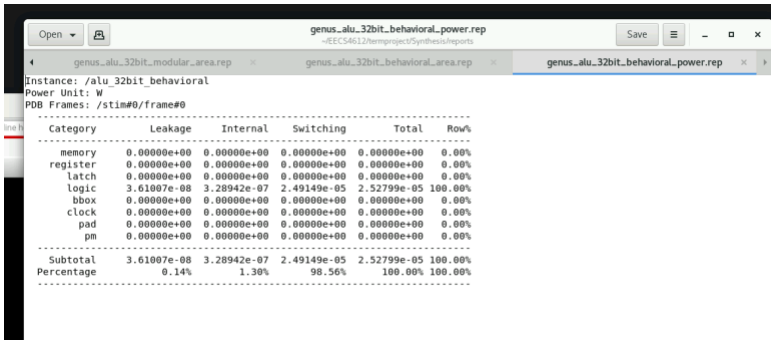
```
*alu_32bit_modular_postRoute_all.tarpt
~/EECS4612/termproject/PnR/timingReports

#############################################################
#  Generated by:      Cadence Innovus 21.17-s075_1
#  OS:                Linux x86_64(Host ID crimson)
#  Generated on:      Sat Dec 13 15:09:43 2025
#  Design:            alu_32bit_modular
#  Command:           timeDesign -postRoute -pathReports -drvReports -slackReports
#############################################################
Path 1: MET Late External Delay Assertion
Endpoint:   Cout   (^) checked with  leading edge of 'clk'
Beginpoint: sel[1] (^) triggered by  leading edge of 'clk'
Path Groups: {clk}
Analysis View: worst_case
Other End Arrival Time          0.000
- External Delay                0.030
+ Phase Shift                1000.000
= Required Time              999.970
- Arrival Time                86.157
= Slack Time                 913.812
     Clock Rise Edge                     0.000
     + Input Delay                       0.010
     + Drive Adjustment                  0.112
     = Beginpoint Arrival Time           0.122
```

The above screenshots show the modular ALU which show an area **of 1451.105** x 10 ^-6 meter squared. The power is **2.5 x 10^-5 watts**. According to the timing analysis, the max frequency of this circuit is: 1/arrival time = 11.61 mhZ.

Now let's compare this to the **behavioral chip**. After that we will compare both stats side by side.



```
genus_alu_32b
~/EECS4612/ter
genus_script.tcl   genus_alu_32bit_modular_power.rep

  Area mode:              timing library
==========================================================


   Gate    Instances   Area       Library
-----------------------------------------------
ADDFX1            32  164.160    slow_vdd1v0
AND2X1             2    2.736    slow_vdd1v0
AO21X1            32   76.608    slow_vdd1v0
AOI22X1           32   65.664    slow_vdd1v0
AOI32X1           32   76.608    slow_vdd1v0
BUFX2             32   54.720    slow_vdd1v0
CLKAND2X8          1    5.814    slow_vdd1v0
CLKBUFX20         33  270.864    slow_vdd1v0
INVX1             35   23.940    slow_vdd1v0
MX2XL             32   76.608    slow_vdd1v0
NAND2X1            1    1.026    slow_vdd1v0
NAND2XL           32   32.832    slow_vdd1v0
NOR2BX2            1    2.736    slow_vdd1v0
NOR2X1             1    1.026    slow_vdd1v0
NOR2X4             1    3.078    slow_vdd1v0
OA21X2             1    3.078    slow_vdd1v0
OAI211X1          64  109.440    slow_vdd1v0
OR2X1              1    1.368    slow_vdd1v0
OR2X2              2    3.420    slow_vdd1v0
-----------------------------------------------
total            367  975.726



     Type      Instances   Area  Area %
-----------------------------------------
inverter              35  23.940    2.5
buffer                65 325.584   33.4
logic                267 626.202   64.2
physical_cells         0   0.000    0.0
-----------------------------------------
total                367 975.726  100.0
```



```
genus_alu_32bit_behavioral_power.rep
~/EECS4612/termproject/Synthesis/reports
genus_alu_32bit_modular_area.rep   genus_alu_32bit_behavioral_area.rep   genus_alu_32bit_behavioral_power.rep

Instance: /alu_32bit_behavioral
Power Unit: W
PDB Frames: /stim#0/frame#0
-----------------------------------------------------------------------------
  Category       Leakage     Internal    Switching       Total    Row%
-----------------------------------------------------------------------------
    memory     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
  register     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
     latch     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
     logic     3.61007e-08  3.28942e-07  2.49149e-05  2.52799e-05 100.00%
      bbox     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
     clock     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
       pad     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
        pm     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
-----------------------------------------------------------------------------
  Subtotal     3.61007e-08  3.28942e-07  2.49149e-05  2.52799e-05 100.00%
Percentage          0.14%        1.30%       98.56%      100.00% 100.00%
-----------------------------------------------------------------------------
```

```
alu_32bit_behavioral_postRoute_all.tarpt
~/EECS4612/termproject/PnR/timingReports

#############################################################
#  Generated by:      Cadence Innovus 21.17-s075_1
#  OS:                Linux x86_64(Host ID crimson)
#  Generated on:      Sat Dec 13 15:42:28 2025
#  Design:            alu_32bit_behavioral
#  Command:           timeDesign -postRoute -pathReports -drvReports -slackReports
#############################################################
Path 1: MET Late External Delay Assertion
Endpoint:   F[31] (v) checked with  leading edge of 'clk'
Beginpoint: S[0]  (^) triggered by  leading edge of 'clk'
Path Groups: {clk}
Analysis View: worst_case
Other End Arrival Time          0.000
- External Delay                0.030
+ Phase Shift                1000.000
= Required Time              999.970
- Arrival Time                61.008
= Slack Time                 938.962
     Clock Rise Edge                     0.000
     + Input Delay                       0.010
     + Drive Adjustment                  0.059
     = Beginpoint Arrival Time           0.069
```

According to the screenshots above, for the behavioural chip: the total area is: **975.726 x 10^-6 meter squared.** The power used is **2.52 x 10^-6 watts.**

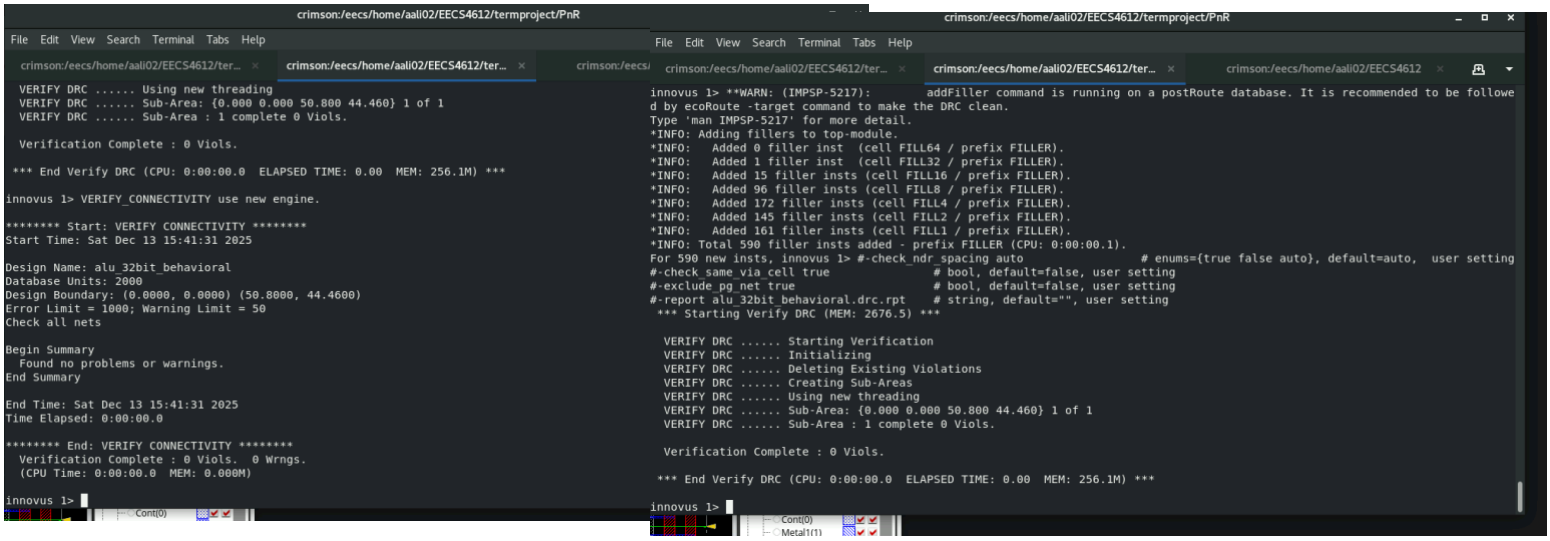
The max frequency can be calculated using 1/arrival time = 16.39 MHZ.

Now here is a table comparing the 2:

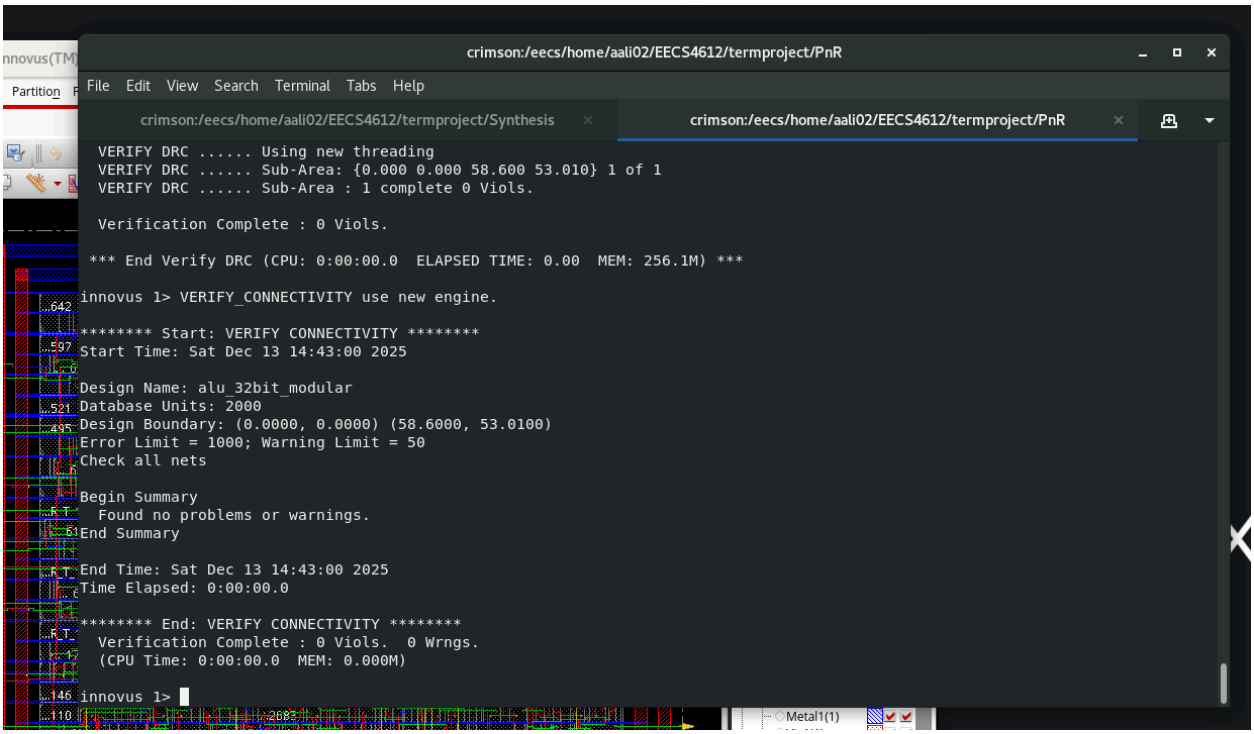
Metric	Modular ALU	Behavioral ALU
Area	1451.106	975.726
Power	2.55 x 10 ^-5	2.52 x 10 ^ -5
Max Frequency	11.61 MHZ	16.39 MHZ

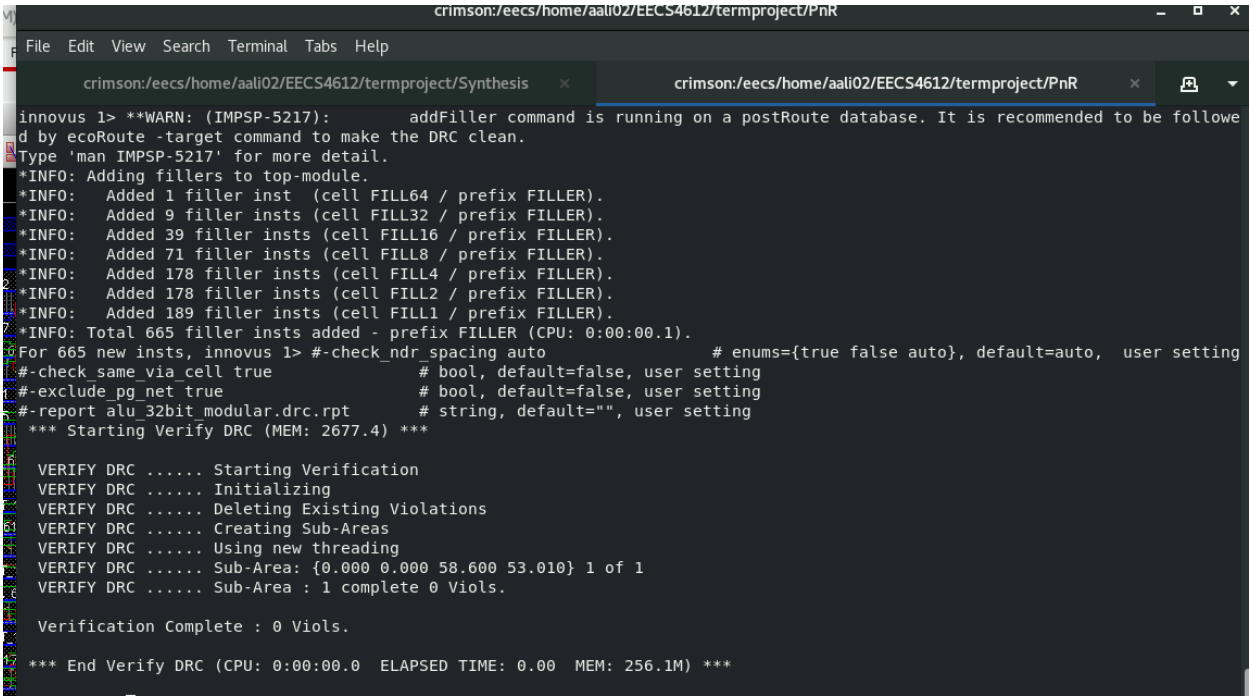
With this, we can see a clear victory for the behavioral ALU which wins in all metrics including Area, power and frequency. Thus, we will use this design moving forward.

DRC and Connectivity for Both ALUs:



This is the DRC and connectivity for the **behavioral** ALU. Both checks passed with 0 violations.

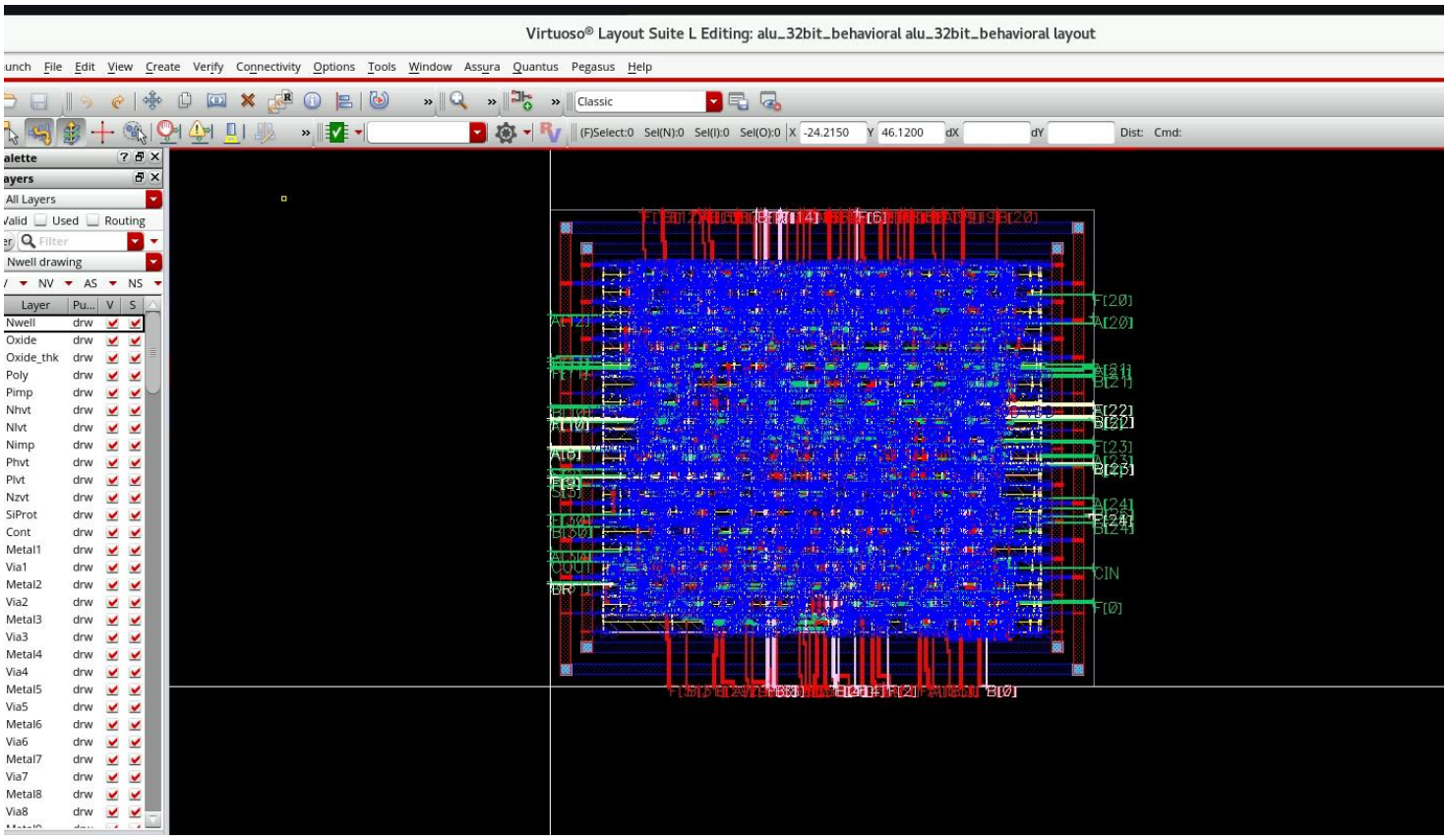




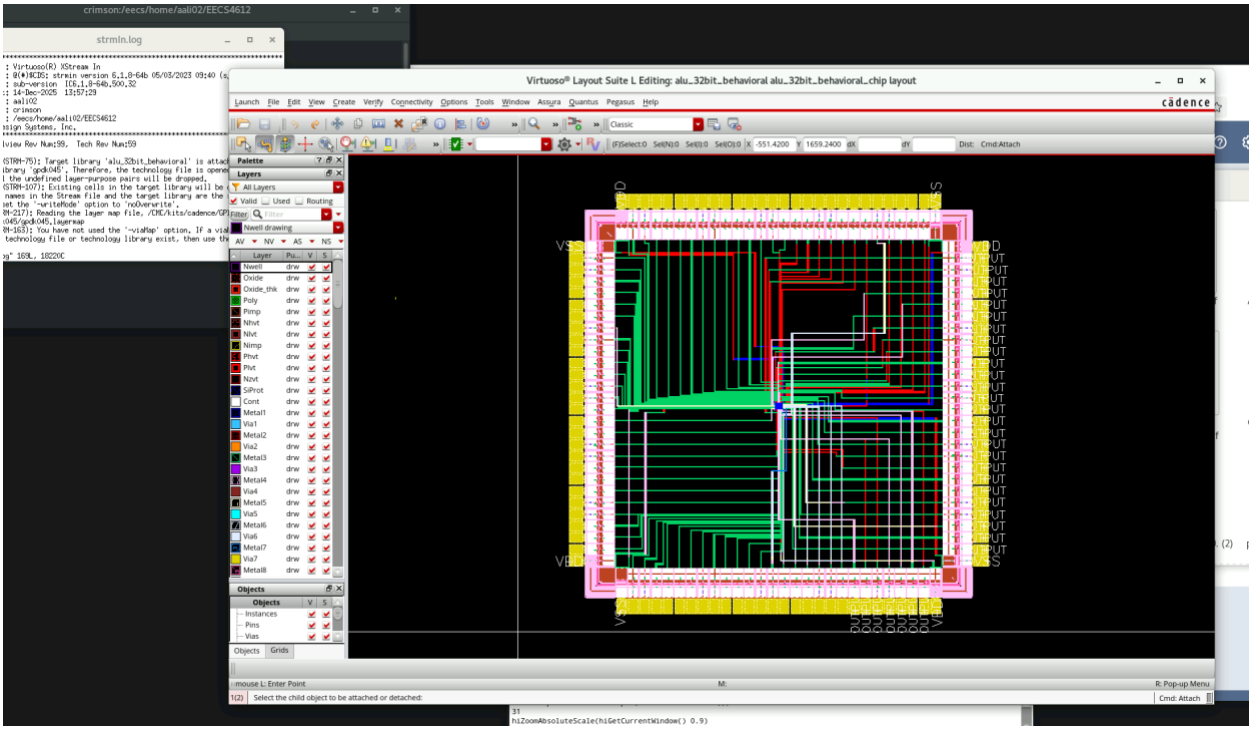
The above screenshots show the DRC and connectivity for the **modular** ALU. For this one the checks passed as well with 0 violations.

Exporting Final Design

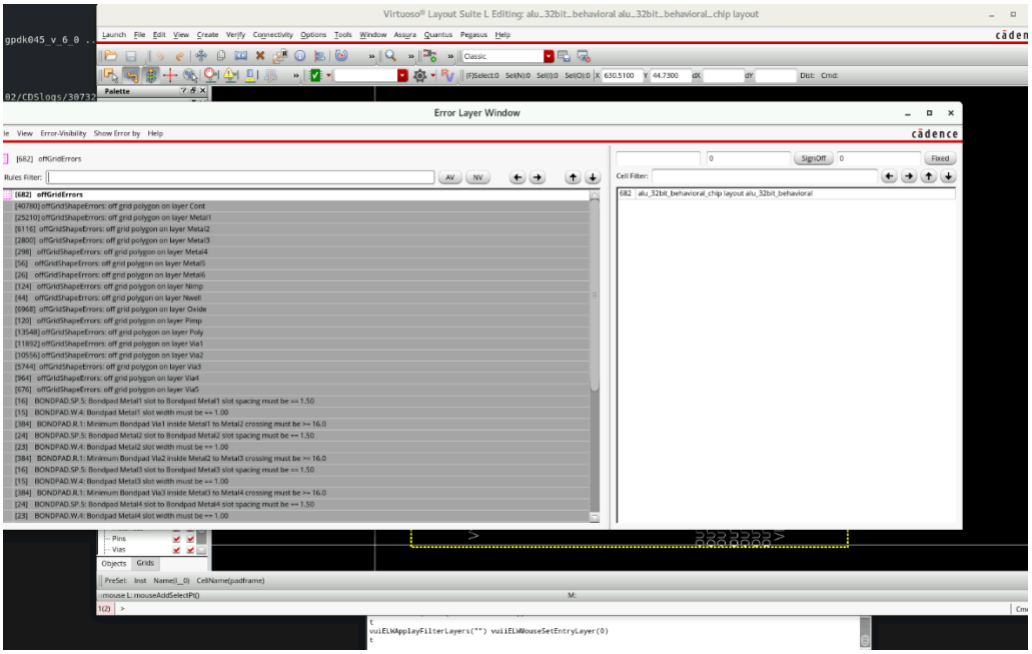
Now that we have decided to move along with the behavioral ALU as our final design, that is the one that we are going to attach to the padframe. The other GDS file for the modular ALU has been uploaded to eClass.



Above is a picture of the final integrated layout of the 32-bit ALU behavioral. This will now be integrated within the padframe.



After instantiating the 32 bit ALU within the padframe, I get this design. I had to manually connect all the pins together and verify the connectivity using DRC. The DRC is shown below. There were some errors below that I simply could not get fixed but I think this is not due to my design and were already there in the padframe.



Above screenshot shows the DRC.

The very last step is the timing analysis. Even though the measurements were conducted, the timing will be the same as the behavioral. We calculate it using the formula $fmax = \frac{1}{criticalpathdelay}$. So, this comes out to be 1/61.008 x 10^-9 = 16.39 MHZ. So, we can conclude that the maximum speed for this 32-bit ALU is 16.39 MHZ.