Final Project
Design and Synthesis of Central Processing Units

Shuhei Aoki
ECE 429-02
Due: April 30, 2018

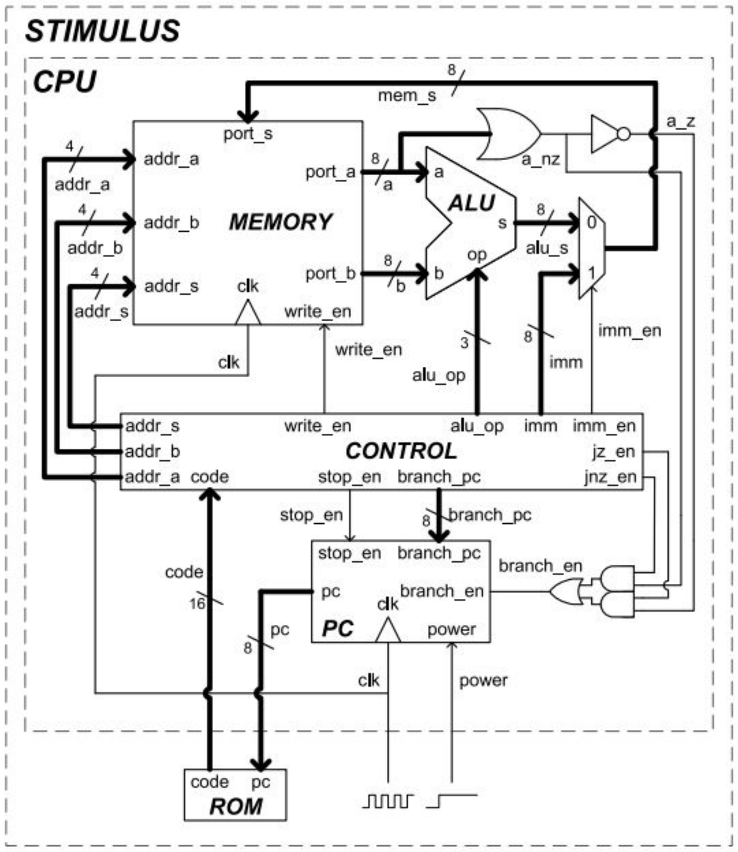
Abstraction

This project is about designing a more efficient 32 bit CPU using a 8 bit CPU as a reference. First, the datalines of the 8 bit CPU was expanded to support 32 bits. Next the adder implementation was changed to a faster implementation. The design process started from register transfer level coded in verilog and finished with a full layout displayed in Virtuoso. The RTL design was then tested through different techniques such as equivalence checking to verify the correctness of the processor.

Introduction

The purpose of the project is to build a 32 bit cpu by using the knowledge obtained through out the labs. Since a CPU is a very complicated combinational logic, it is prone to errors through its designing process. To solve this issue, at each step of the design process, the design will be tested for its functionality. During the synthesizing process, each result will be run through verilog to verify that the product is correct. Additionally, after place and route, the circuit will go through equivalence testing to further confirm the correctness. The design challenge includes modifying a sample 8 bit CPU to create the 32 bit CPU and to implement a faster adder. The challenge can be approached by first expanding the CPU components to support 32 bit data lines. Then modify the CPU for it to use the modify parts. Lastly, modify the adder logic to a carry lookahead adder[1].

Background



CPU with Test Bench[1]

A CPU is a complicated combinational circuit which can be split up into many subcomponents which supports each other. The CPU starts at the PC unit. The PC is responsible to handle the pointer of the current executing instruction. The pointer will goto the next line (unless it branches/jumps) at every clock cycle. This means that the CPU needs to stabilize before the next clock ticks. The PC unit also accepts a branch address and a signal to tell the unit whether or not to overwrite the current PC with the branch PC.

The PC will then fetch the instruction from a ROM and provide it to the control logic which controls all of the components in the CPU. An additional responsibility for the control logic is to decode the instruction provided by the ROM. The format of the instruction will differ depending on the type of instruction. If the instruction is a arithmetic instruction such as adding or ANDing, the control unit will decode the source, and destination memory address and tell the ALU what operation it should use. If the type of instruction is an immediate instruction, the control unit will output the immediate value provided by the instruction. Lastly, if the instruction is a jump instruction, the control will decode the jump instruction and tell the memory to fetch the value provided at the memory location to check whether or not the jump condition is met. In this processor, the branch condition is calculated by oring all of the outputs of a value. If the value is 0, the or will output 0. Otherwise, the output will be 1.

The ALU or arithmetic logic unit has a collection of mathematical operations and the type of operation is chosen by the control logic. The ALU contains the adder operation which has a variety of performance depending on the implementation. Two types of adder, carry ripple adder and carry lookahead adder is explored in the project. The output of the ALU is then connected to a MUX which chooses the output signal depending on the signal input (coming from the control unit). Lastly, the output of the MUX is fed directly back into the memory which can be read from or written to. The CPU can be implemented using the standard cell based ASIC design flow.

The standard cell based ASIC design flow can automatically create a layout from a register-transistor level design. The design flow works by having a library of standard cells which can be combined to create a more complex circuit. The design process starts from synthesis, which the program compiles a register transfer level description into a netlist of logical components specific to the design process such as AND gates, and OR gates. The standard cell libraries have multiple implementations of the same logic but with different properties such as transistor width to optimize for different cases. The last step is to determine the physical placement of each cells and the routing of the metal to connect to the correct cells[4].

At each step, the output was run through tests to verify the correctness of the products. This is done by using test benches and equivalent testing. First, the RTL description is tested by running verilog and displaying the result in simvision. For each rising clock, the CPU should compute of line of instruction. In the last step, the layout which contains the standard cells and the routing between the cells will be compared with the RTL design using equivalence checking in Formality.

32-bit CPU Implementation

The 32 bit CPU was implemented using verilog and a sample 8 bit CPU. The provided 8 bit CPU had all of the data lines set as 8 bits. The components of the 8 bit CPU was first modified to support the 32 bit wide data. However the instruction and the PC address was kept the same as 16 bits and 8 bits respectively. The components were modified in the following order:

- Tristate buffer [Used in memory module]
- D flip flop [Used in memory module]
- Mux (2 to 1) [Used in multiple occasions such as selecting dataline into memory between immediate value and ALU output]
- Memory [Used in memory bank]
- Memory bank (of 16 words)
- AND [Used in ALU]
- XNOR [Used in ALU]
- Adder (Including the add subtraction wrapper)[Used in ALU]
- Shifter [Used in ALU]
- ALU [Used in 32 bit CPU]
- Control [Used in 32 bit CPU]
- 32 Bit CPU

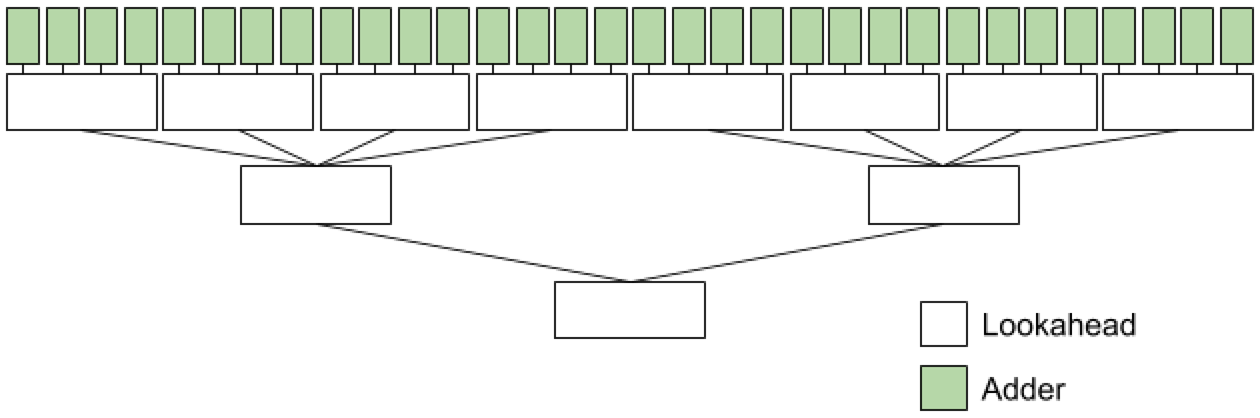
Since the more complex modules are built on top of less complex modules, the underlying modules were first converted to 32 bit first. Then the more complicated modules (such as the ALU) were modified to support 32 bit.

The process of converting the modules to 32 bit was done by modifying the input, output and internal data lines to 32 bits. This means that additional components and wiring ([7:0] to [31:0]) were introduced to each module. For example, for the 32 bit d flip flop, 24 single bit d flip flops were added to hold 32 bits of data. Additionally, for the more complex modules (which contained other modules), the 8 bit modules used internally were replaced with 32 bit counter parts. For the control unit, only the immediate value output needed to be changed. The 8 bit immediate value obtained from the instruction was bit extended to 32 bits.

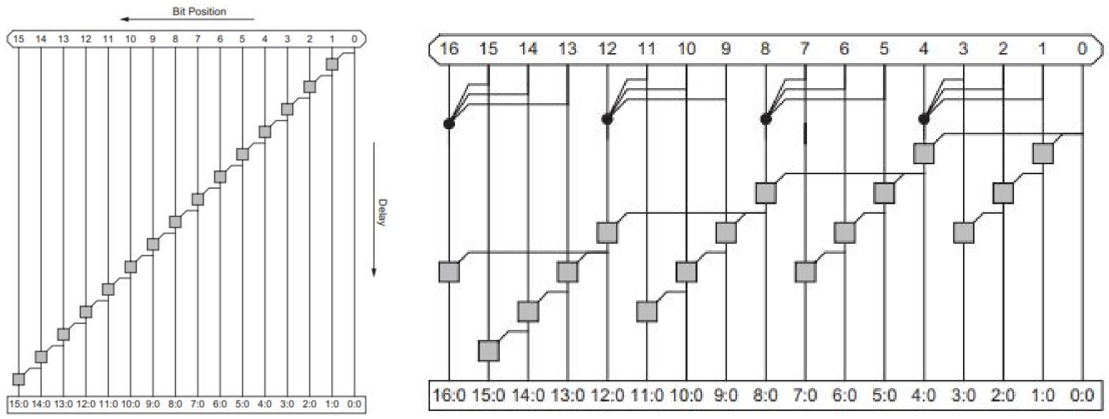
Architectural Exploration of Adders

The 8 bit adder implemented the carry ripple adder (CRA). The CRA implementation was then carried over to the 32 bit CPU. The carry ripple works by connecting a full bit adder in series where the carry out of one full adder is connected to the carry in of the next. This is the most simple and straightforward implementation of an adder. However, this creates a long ripple of carry signal which makes the critical path very long (as seen in the figure below). In order for the addition to finish computing, the previous full adder needs to be completed in order for the next adder to finish stabilizing. This means that for a 32 bit CRA, the whole array of full adders needs to finish stabilize in sequence in order for the carry out and sum to be correct.

The carry lookahead adder (CLA) improves on CRA by calculating the generate and propagate signals in a group. This eliminates the need for each adder to wait for the ripple. The CLA was implemented in this project by having 4 bit lookahead modules and connecting it in tree like structure by having superblocks which connects 4 lookahead modules and another which connects 2 lookahead modules (seen in the figure below)[3]. However, since this adds more wiring and gates, the additional delay might make the CLA perform worse than the CRA in the worst case scenario[1].



Hierarchical Structure of CLA



Comparison between Timing of RCA (left) and CLA (right)[3]

Functional Validation and Verification

During the designing process of the CPU, it went through multiple verifications to confirm that the CPU has the same functionality at each step of the process. The first verification method that was used was running the simulation and comparing the test bench output with the correct answer (which can be seen in the post simulation results in appendix). Running a sample test program like this verifies the correctness of the CPU because it tested most of the possible instructions (sll, add, sub, loadi, jz, jnz, and stop). Since the test bench code used in the testing was multiplication, the initial test consisted of comparing the result of the test bench with the calculated product of

$$4111 * 332606 = 1367343266$$

The designing phase started from the register-transfer level (RTL). After the RTL design was completed, it was run through an simulation and the returned product was checked. This step was repeated with post design compiler. Lastly, at the end of the design process, after place and route, the output is run through equivalence checking using software instead of checking by the user.

The last equivalence checking is done through formality which it compares the functionality of the original RTL design and post place and route layout. The software first asks for the reference design (RTL) and the implementation (post place and route). Then it will automatically, match the ports between the two design and checks if the functionality is identical.

Synthesis Results

First step after completing the RTL description, the design will go through logic synthesis. In this step, all of the gates and modules are replaced by into standard cells. The result is written to cpu32.vh and contains a netlist of interconnected standard cells. The top level of the module and the frequency is set in the configuration file before compilation. The compiler also is able to estimate metadata of the cell, power and timing of the circuit. After, the result will be sent to place and route using encounter. This step places and routes the gates into a certain position. Since the placement of the cells are determined now, the gates can be wired up with physical distance of the metal interconnects. Because of that, now the timing, power, and cell estimates are more closer to real life since it takes into account the power, and delay caused by the interconnect. Encounter will then compile the result to final.v. The last step is to import the design into virtuoso which completes the design process by filling the layout with wells and poly[2].

The results of the simulation are seen in the table below. The table shows the comparison between the metadata of the cpu (area, power and timing) of different adder implementation and results between different steps. When comparing the area, the area increases after running place and route. However, it seems that the place and route compiler optimizes the circuit because the total surface area is decreased even though the number of cells increased. Additionally, when comparing the power and timing between post logic synthesis/Design Compiling(DC) and place and route (PR), post PR has a larger power consumption with a larger delay. This comes from the introduction of interconnects post PR.

When comparing the performance of the two adder, one can see that the area, power, and worst case timing increases from the table below. However, if the circuit is not running on a worst case scenario, CLA is faster than CRA. This shows that the carry lookahead adder is faster than carry ripple adder if its not on its worst condition. The lower worst case scenario comes from the additional circuit required when building the faster circuit. Since there are more components, the area increases, and the delay increases proportionally as well[1].

		CRA			**CLA**		
Measurement		**Post DC**	**Post PR**		**Post DC**	**PostPR**	
	Cells	4906	4958		4978	5030	
Area	**Total Area (um)**	15368.6	13800.2		15546.03	13971.1	
	Internal Power	1.088	2.227		1.105	2.274	
	Switching Power	0.169	1.711		0.177	1.729	
	Leakage Power	0.06673	0.06801		0.06811	0.06939	
Power (mw)	**Total Power**	1.32373	4.00601		1.35011	4.07239	
	Required Time	7.9	7.055		7.9	7.56	
	Arrival TIme	-4.62	7.525	7.304	-3.61	7.746	7.515
Timing (ns)	**Slack**	3.28	-0.47	-0.249	4.29	-0.186	0.046
			Worst	**10th**		**Worst**	**10th**

Conclusion and Future Work

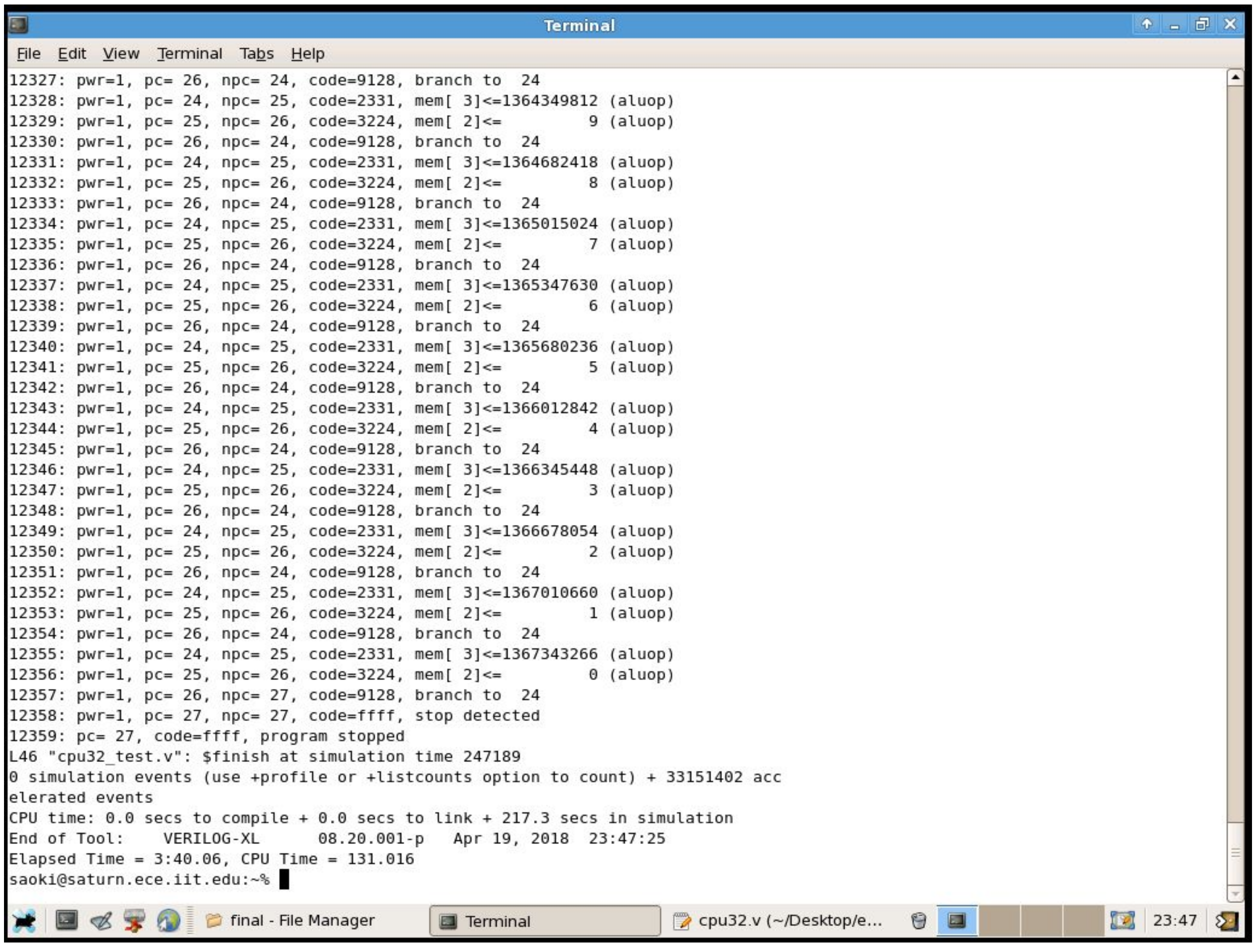
Students were able to learn about CPU and the standard design process with the project. First, the cpu was expanded into 32 bits, then the adder was optimized for better performance. Next, the cpu went through testing to verify its functionality. However, this is only scratching the surface of a modern cpu design. There are more components needed to support more standard instructions such as hardware implemented multiplication and division and floating point operations. Additionally, each components can be further optimized such as finding a faster implementation of an adder or finding a way to use less gates to perform the same operations.

References

[1] Final Project Instructions
[2] Tutorial IV: Standard Cell Based ASIC Design Flow
[3] Weste, Neil H. E., and David Money. Harris. CMOS VLSI Design: a Circuits and Systems Perspective. 4th ed., Addison Wesley, 2011.
[4] Lab 9: Standard Cell Based ASIC Design Flow

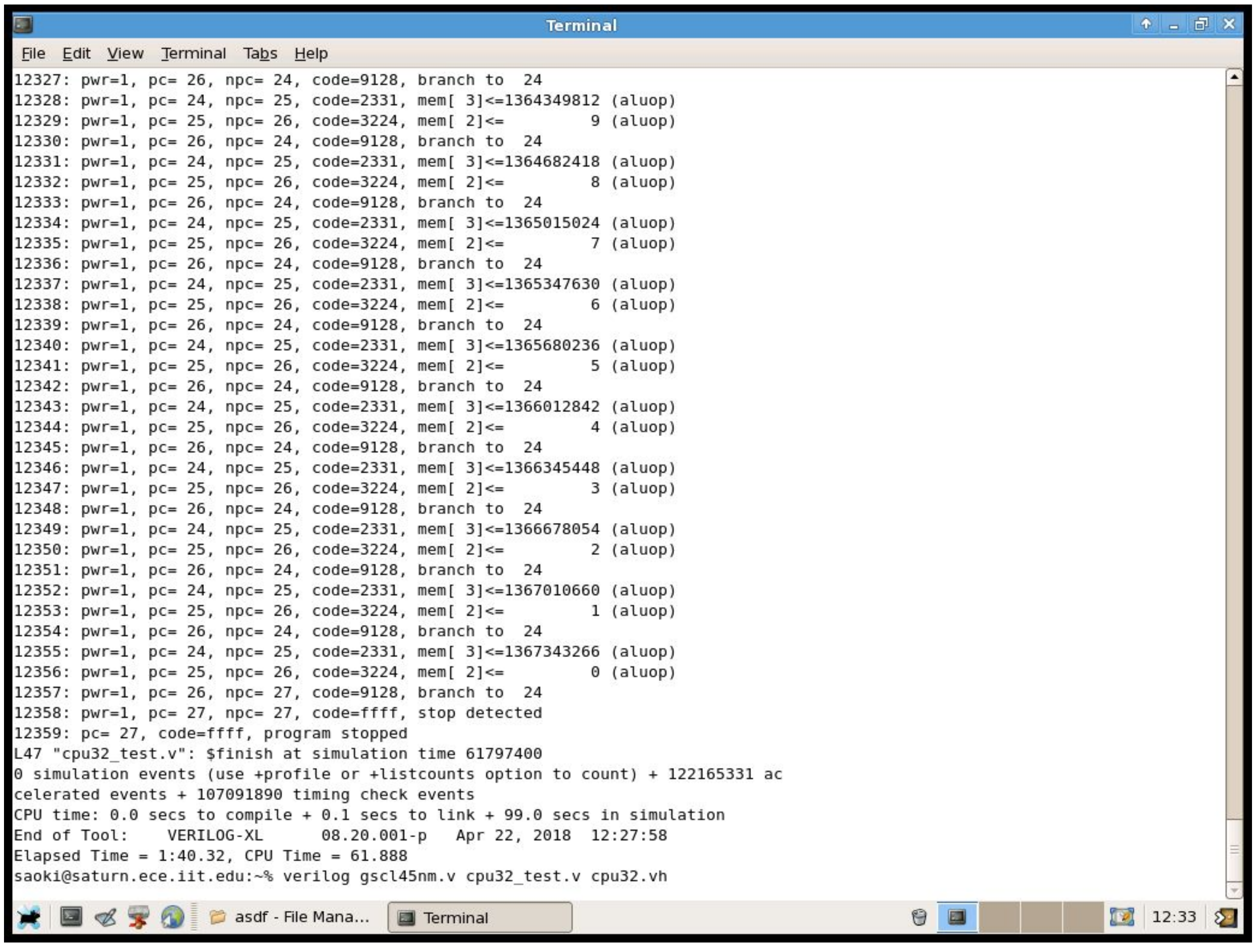
Appendix

1. Simulation Result of RTL Description for a 32 Bit CPU with CRA
2. Post DC Simulation Result for a 32 Bit CPU with CRA
3. Post Place and Route Simulation Result for a 32 Bit CPU with CRA
4. Equivalence Checking for 32 Bit CPU with CRA
5. Layout of a 32 Bit CPU with CRA
6. Simulation Result of RTL Description for a 32 Bit CPU with CLA
7. Post DC Simulation Result for a 32 Bit CPU with CLA
8. Post Place and Route Simulation Result for a 32 Bit CPU with CLA
9. Equivalence Checking for 32 Bit CPU with CLA
10. Layout of a 32 Bit CPU with CLA



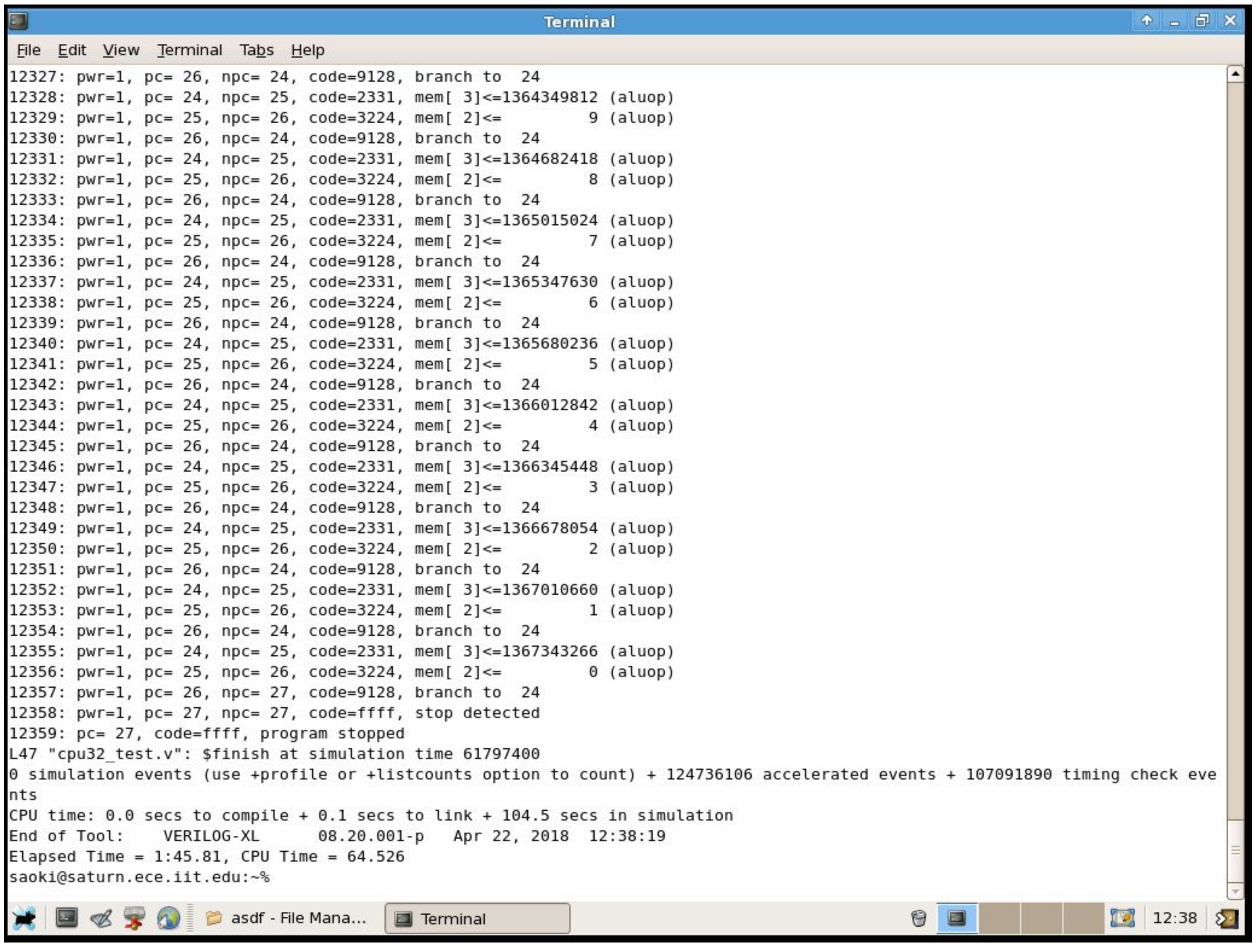
```
12327: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12328: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1364349812 (aluop)
12329: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         9 (aluop)
12330: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12331: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1364682418 (aluop)
12332: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         8 (aluop)
12333: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12334: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1365015024 (aluop)
12335: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         7 (aluop)
12336: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12337: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1365347630 (aluop)
12338: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         6 (aluop)
12339: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12340: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1365680236 (aluop)
12341: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         5 (aluop)
12342: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12343: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1366012842 (aluop)
12344: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         4 (aluop)
12345: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12346: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1366345448 (aluop)
12347: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         3 (aluop)
12348: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12349: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1366678054 (aluop)
12350: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         2 (aluop)
12351: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12352: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1367010660 (aluop)
12353: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         1 (aluop)
12354: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12355: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1367343266 (aluop)
12356: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         0 (aluop)
12357: pwr=1, pc= 26, npc= 27, code=9128, branch to  24
12358: pwr=1, pc= 27, npc= 27, code=ffff, stop detected
12359: pc= 27, code=ffff, program stopped
L46 "cpu32_test.v": $finish at simulation time 247189
0 simulation events (use +profile or +listcounts option to count) + 33151402 acc
elerated events
CPU time: 0.0 secs to compile + 0.0 secs to link + 217.3 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Apr 19, 2018  23:47:25
Elapsed Time = 3:40.06, CPU Time = 131.016
saoki@saturn.ece.iit.edu:~%
```

Figure 1: Simulation Result of RTL Description for a 32 Bit CPU with CRA



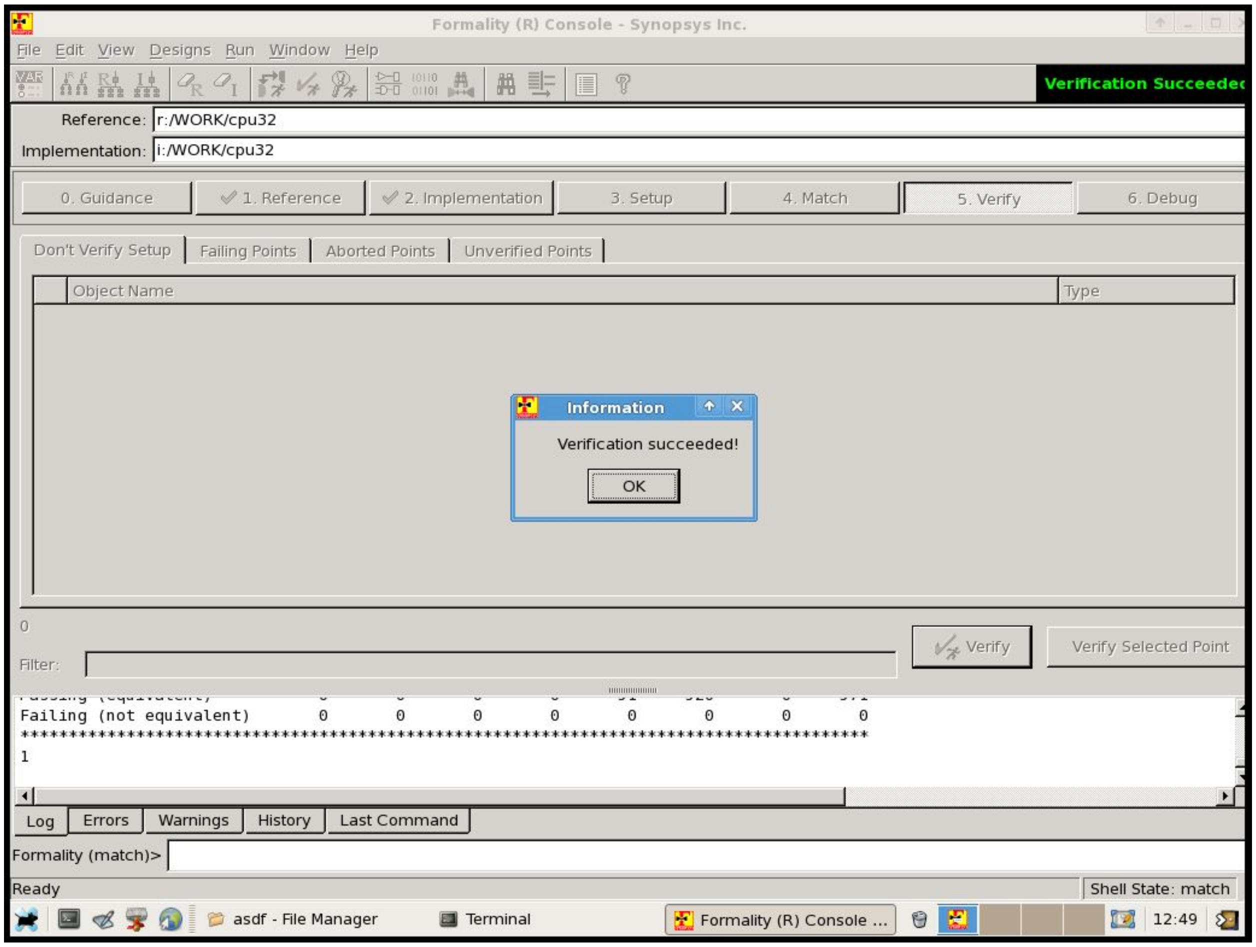
```
12327: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12328: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1364349812 (aluop)
12329: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         9 (aluop)
12330: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12331: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1364682418 (aluop)
12332: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         8 (aluop)
12333: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12334: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1365015024 (aluop)
12335: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         7 (aluop)
12336: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12337: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1365347630 (aluop)
12338: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         6 (aluop)
12339: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12340: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1365680236 (aluop)
12341: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         5 (aluop)
12342: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12343: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1366012842 (aluop)
12344: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         4 (aluop)
12345: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12346: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1366345448 (aluop)
12347: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         3 (aluop)
12348: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12349: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1366678054 (aluop)
12350: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         2 (aluop)
12351: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12352: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1367010660 (aluop)
12353: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         1 (aluop)
12354: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12355: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1367343266 (aluop)
12356: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         0 (aluop)
12357: pwr=1, pc= 26, npc= 27, code=9128, branch to  24
12358: pwr=1, pc= 27, npc= 27, code=ffff, stop detected
12359: pc= 27, code=ffff, program stopped
L47 "cpu32_test.v": $finish at simulation time 61797400
0 simulation events (use +profile or +listcounts option to count) + 122165331 ac
celerated events + 107091890 timing check events
CPU time: 0.0 secs to compile + 0.1 secs to link + 99.0 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Apr 22, 2018  12:27:58
Elapsed Time = 1:40.32, CPU Time = 61.888
saoki@saturn.ece.iit.edu:~% verilog gscl45nm.v cpu32_test.v cpu32.vh
```

Figure 2: Post DC Simulation Result for a 32 Bit CPU with CRA



```
12327: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12328: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1364349812 (aluop)
12329: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         9 (aluop)
12330: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12331: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1364682418 (aluop)
12332: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         8 (aluop)
12333: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12334: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1365015024 (aluop)
12335: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         7 (aluop)
12336: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12337: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1365347630 (aluop)
12338: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         6 (aluop)
12339: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12340: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1365680236 (aluop)
12341: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         5 (aluop)
12342: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12343: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1366012842 (aluop)
12344: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         4 (aluop)
12345: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12346: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1366345448 (aluop)
12347: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         3 (aluop)
12348: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12349: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1366678054 (aluop)
12350: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         2 (aluop)
12351: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12352: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1367010660 (aluop)
12353: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         1 (aluop)
12354: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12355: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1367343266 (aluop)
12356: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         0 (aluop)
12357: pwr=1, pc= 26, npc= 27, code=9128, branch to  24
12358: pwr=1, pc= 27, npc= 27, code=ffff, stop detected
12359: pc= 27, code=ffff, program stopped
L47 "cpu32_test.v": $finish at simulation time 61797400
0 simulation events (use +profile or +listcounts option to count) + 124736106 accelerated events + 107091890 timing check eve
nts
CPU time: 0.0 secs to compile + 0.1 secs to link + 104.5 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Apr 22, 2018  12:38:19
Elapsed Time = 1:45.81, CPU Time = 64.526
saoki@saturn.ece.iit.edu:~%
```

Figure 3: Post Place and Route Simulation Result for a 32 Bit CPU with CRA



Reference: r:/WORK/cpu32

Implementation: i:/WORK/cpu32

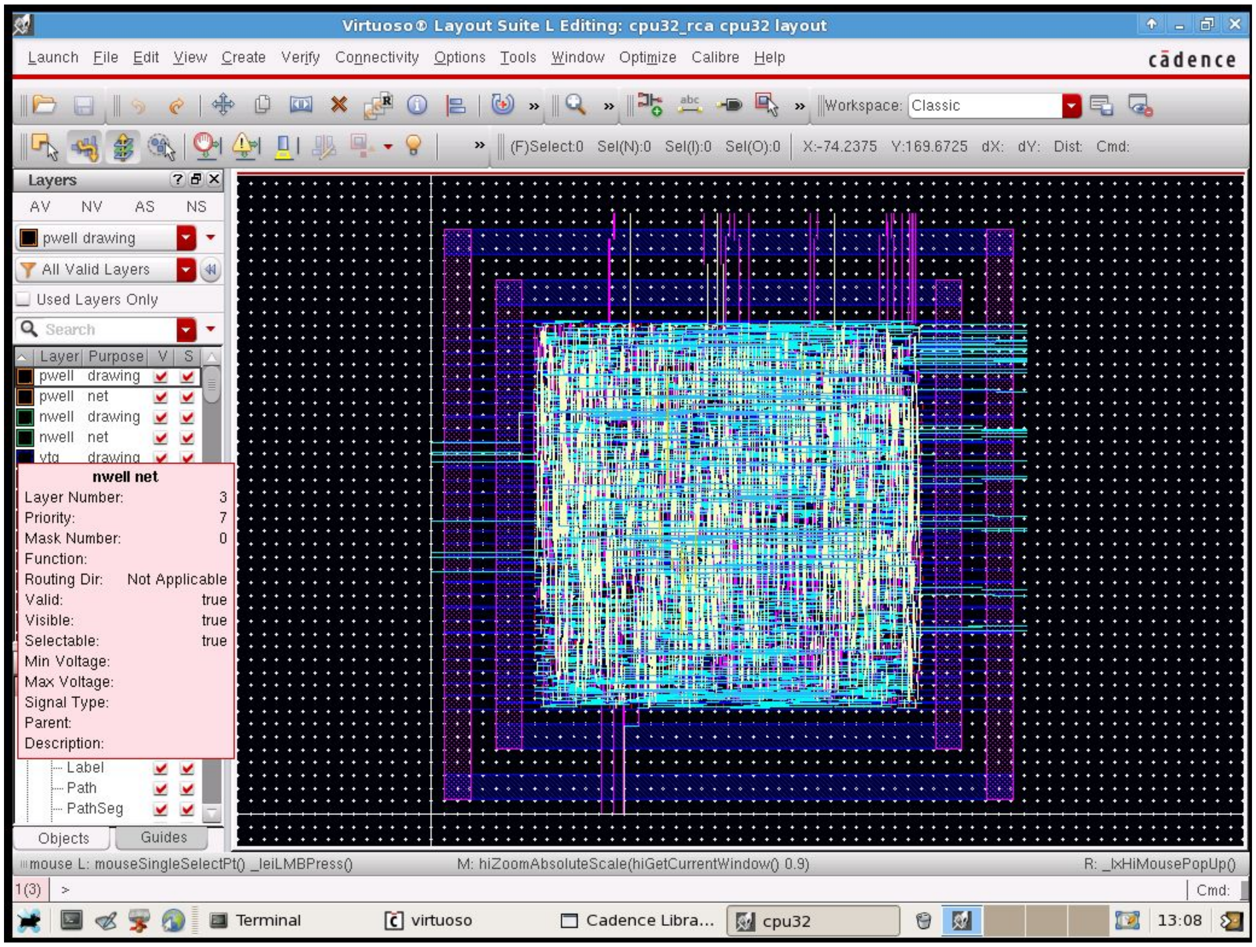
Verification succeeded!

```
Failing (not equivalent)       0       0       0       0       0       0       0       0
****************************************************************************************
1
```

Formality (match)>

Shell State: match

Figure 4: Equivalence Checking for 32 Bit CPU with CRA


Virtuoso® Layout Suite L Editing: cpu32_rca cpu32 layout
Launch File Edit View Create Verify Connectivity Options Tools Window Optimize Calibre Help
Workspace: Classic
(F)Select:0 Sel(N):0 Sel(I):0 Sel(O):0 X:-74.2375 Y:169.6725 dX: dY: Dist: Cmd:
Layers
pwell drawing
All Valid Layers
Used Layers Only
Objects
Guides
Terminal
virtuoso
Cadence Libra...
cpu32
13:08

Layer | Purpose | V | S
pwell drawing ✓ ✓
pwell net ✓ ✓
nwell drawing ✓ ✓
nwell net ✓ ✓
vtg drawing ✓ ✓

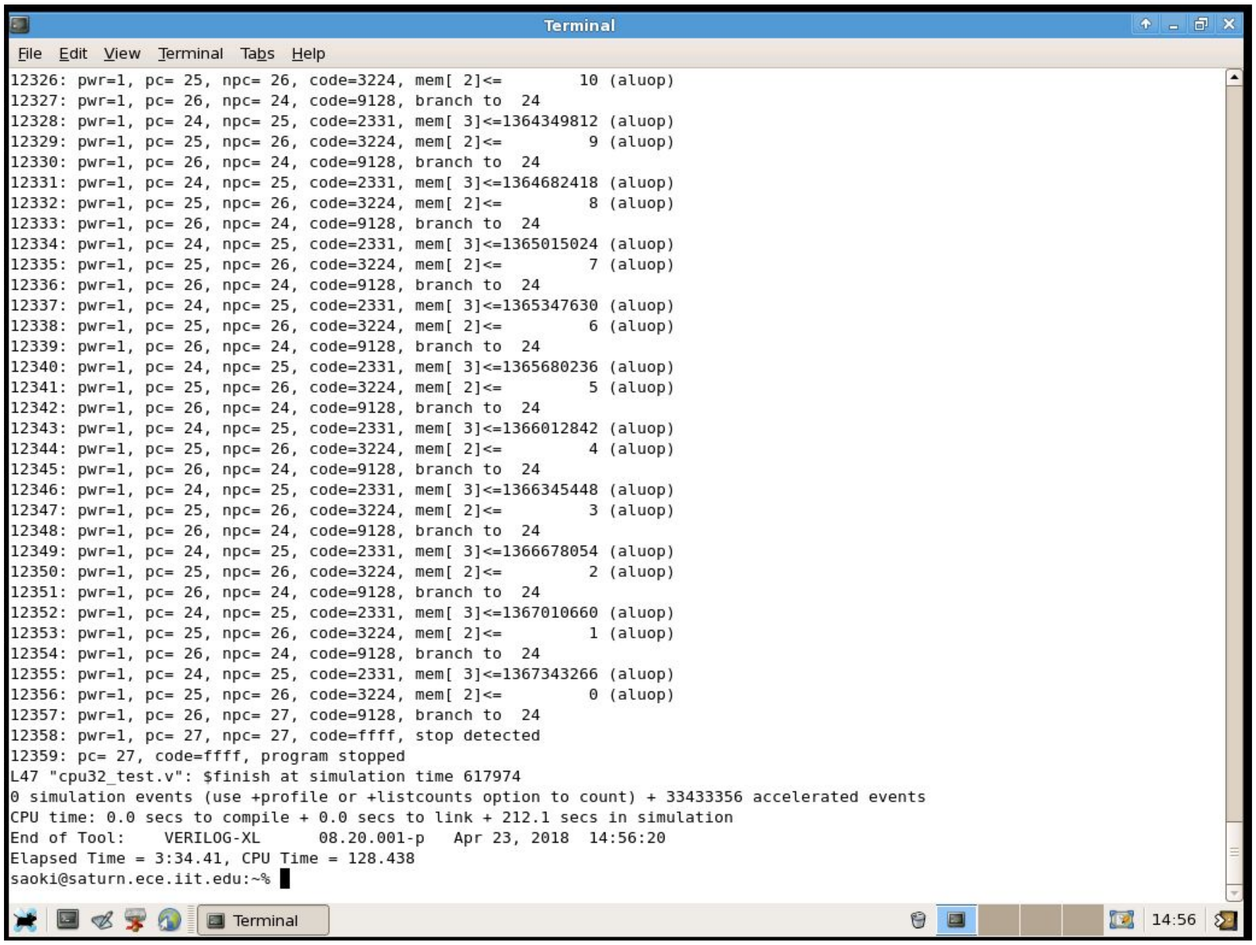
nwell net
Layer Number: 3
Priority: 7
Mask Number: 0
Function:
Routing Dir: Not Applicable
Valid: true
Visible: true
Selectable: true
Min Voltage:
Max Voltage:
Signal Type:
Parent:
Description:

Label ✓ ✓
Path ✓ ✓
PathSeg ✓ ✓

mouse L: mouseSingleSelectPt() _leiLMBPress() M: hiZoomAbsoluteScale(hiGetCurrentWindow() 0.9) R: _lxHiMousePopUp()

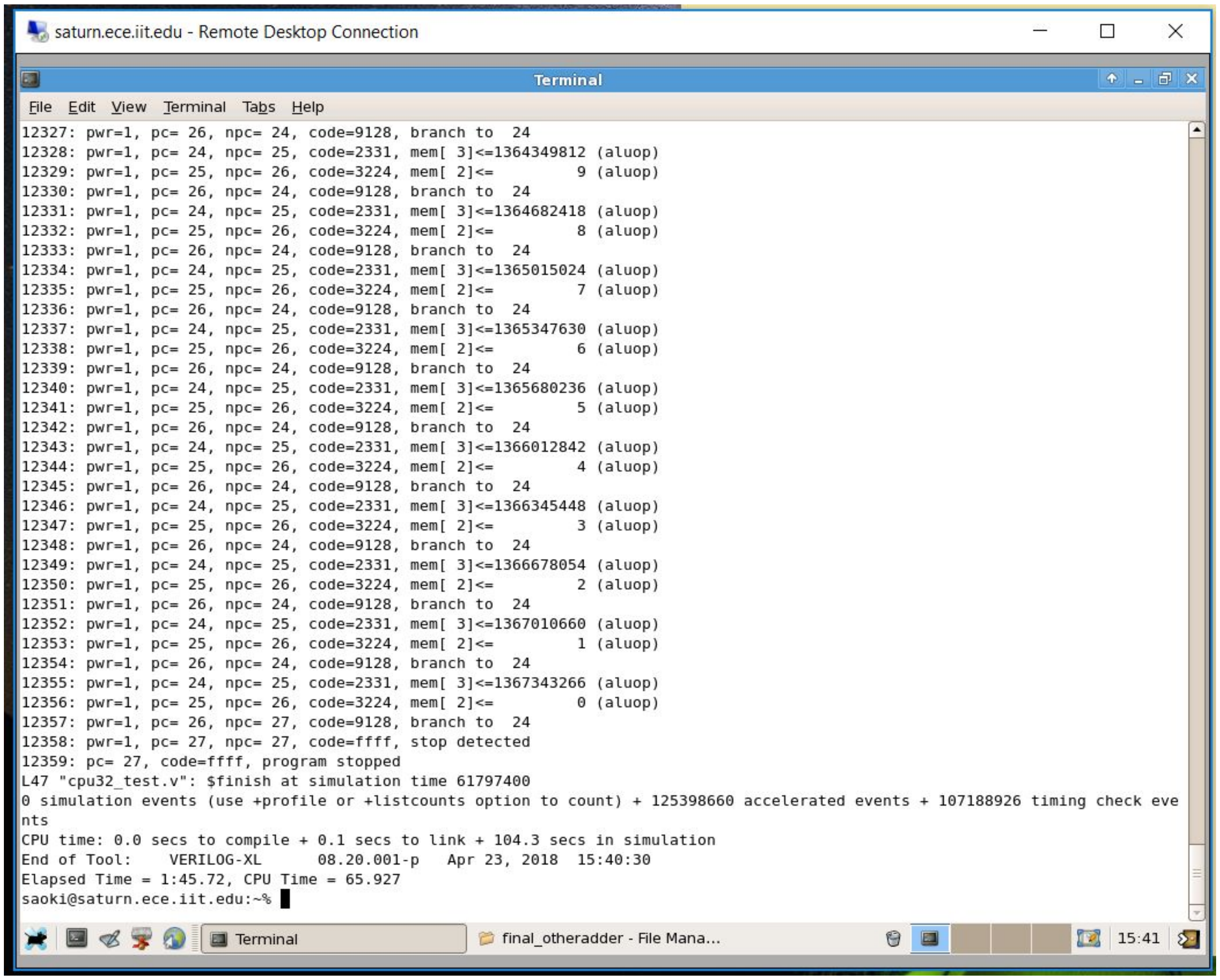
1(3) > Cmd:

Figure 5: Layout of a 32 Bit CPU with CRA



```
12326: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=        10 (aluop)
12327: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12328: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1364349812 (aluop)
12329: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         9 (aluop)
12330: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12331: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1364682418 (aluop)
12332: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         8 (aluop)
12333: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12334: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1365015024 (aluop)
12335: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         7 (aluop)
12336: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12337: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1365347630 (aluop)
12338: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         6 (aluop)
12339: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12340: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1365680236 (aluop)
12341: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         5 (aluop)
12342: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12343: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1366012842 (aluop)
12344: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         4 (aluop)
12345: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12346: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1366345448 (aluop)
12347: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         3 (aluop)
12348: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12349: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1366678054 (aluop)
12350: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         2 (aluop)
12351: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12352: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1367010660 (aluop)
12353: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         1 (aluop)
12354: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12355: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1367343266 (aluop)
12356: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         0 (aluop)
12357: pwr=1, pc= 26, npc= 27, code=9128, branch to  24
12358: pwr=1, pc= 27, npc= 27, code=ffff, stop detected
12359: pc= 27, code=ffff, program stopped
L47 "cpu32_test.v": $finish at simulation time 617974
0 simulation events (use +profile or +listcounts option to count) + 33433356 accelerated events
CPU time: 0.0 secs to compile + 0.0 secs to link + 212.1 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Apr 23, 2018  14:56:20
Elapsed Time = 3:34.41, CPU Time = 128.438
saoki@saturn.ece.iit.edu:~%
```

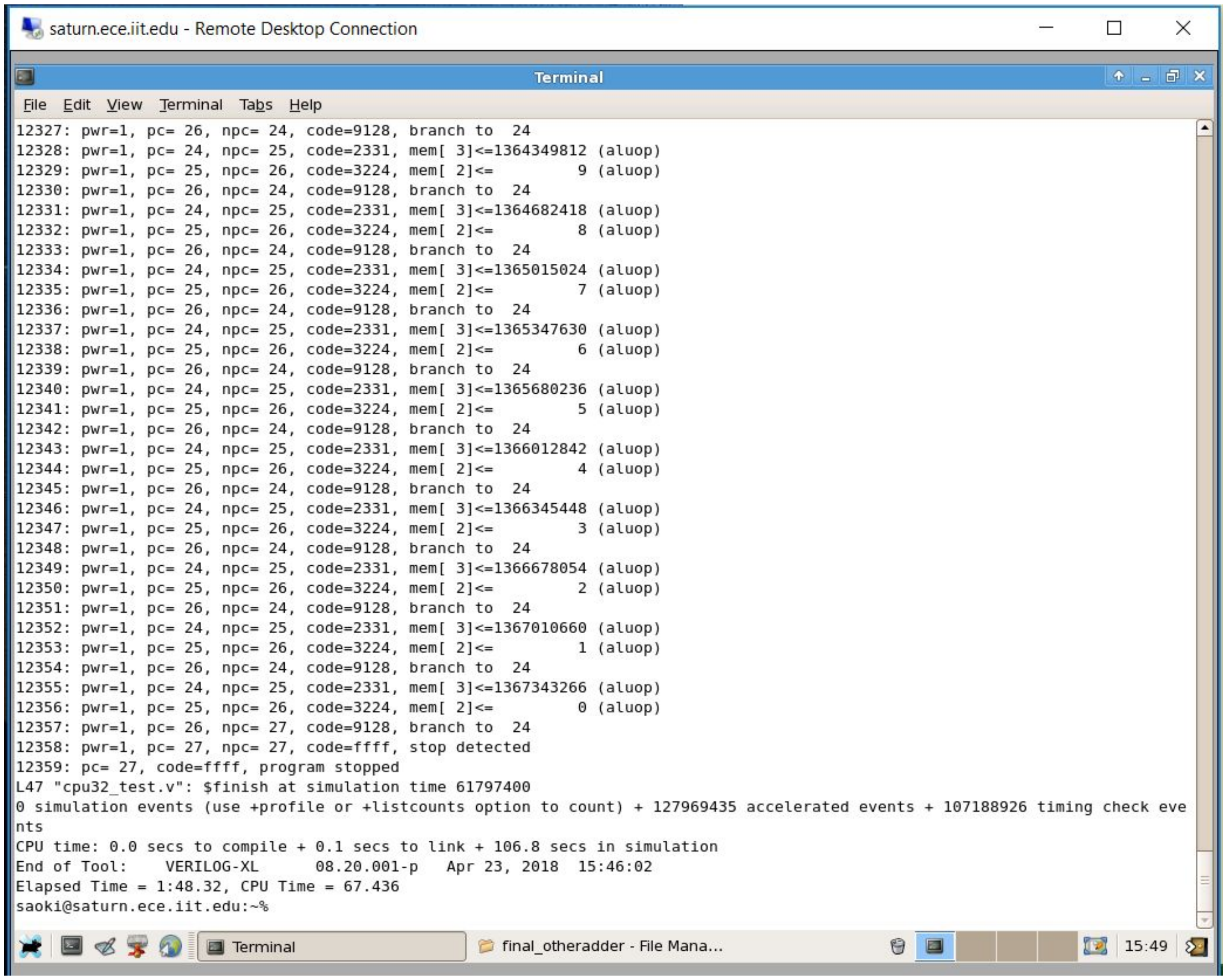
Figure 6: Simulation Result of RTL Description for a 32 Bit CPU with CLA



saturn.ece.iit.edu - Remote Desktop Connection
Terminal
File Edit View Terminal Tabs Help
Terminal
final_otheradder - File Mana...
15:41

```
12327: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12328: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1364349812 (aluop)
12329: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         9 (aluop)
12330: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12331: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1364682418 (aluop)
12332: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         8 (aluop)
12333: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12334: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1365015024 (aluop)
12335: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         7 (aluop)
12336: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12337: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1365347630 (aluop)
12338: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         6 (aluop)
12339: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12340: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1365680236 (aluop)
12341: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         5 (aluop)
12342: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12343: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1366012842 (aluop)
12344: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         4 (aluop)
12345: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12346: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1366345448 (aluop)
12347: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         3 (aluop)
12348: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12349: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1366678054 (aluop)
12350: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         2 (aluop)
12351: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12352: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1367010660 (aluop)
12353: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         1 (aluop)
12354: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12355: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1367343266 (aluop)
12356: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         0 (aluop)
12357: pwr=1, pc= 26, npc= 27, code=9128, branch to  24
12358: pwr=1, pc= 27, npc= 27, code=ffff, stop detected
12359: pc= 27, code=ffff, program stopped
L47 "cpu32_test.v": $finish at simulation time 61797400
0 simulation events (use +profile or +listcounts option to count) + 125398660 accelerated events + 107188926 timing check eve
nts
CPU time: 0.0 secs to compile + 0.1 secs to link + 104.3 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Apr 23, 2018  15:40:30
Elapsed Time = 1:45.72, CPU Time = 65.927
saoki@saturn.ece.iit.edu:~%
```

Figure 7: Post DC Simulation Result for a 32 Bit CPU with CLA

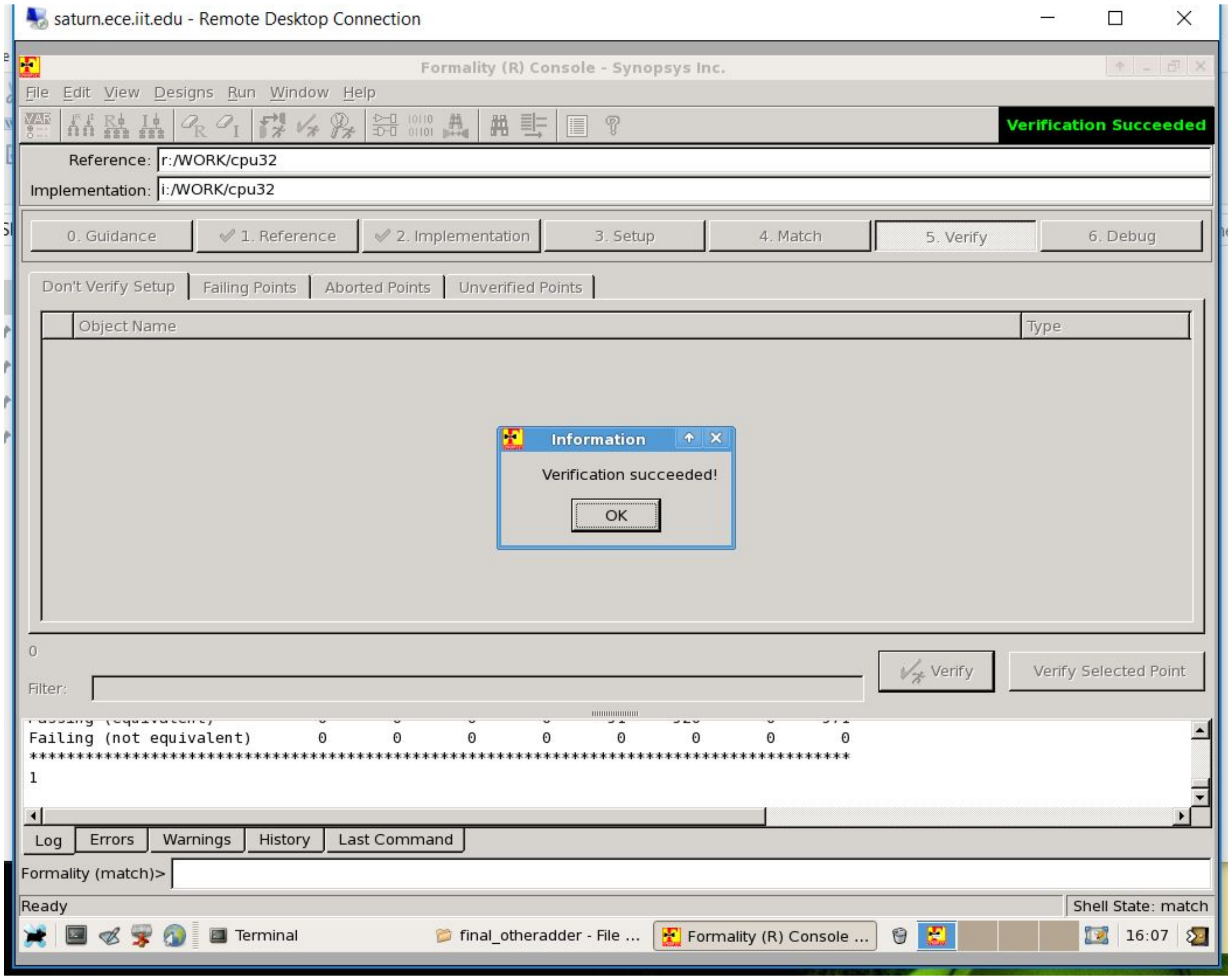


saturn.ece.iit.edu - Remote Desktop Connection

```
12327: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12328: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1364349812 (aluop)
12329: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         9 (aluop)
12330: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12331: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1364682418 (aluop)
12332: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         8 (aluop)
12333: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12334: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1365015024 (aluop)
12335: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         7 (aluop)
12336: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12337: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1365347630 (aluop)
12338: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         6 (aluop)
12339: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12340: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1365680236 (aluop)
12341: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         5 (aluop)
12342: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12343: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1366012842 (aluop)
12344: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         4 (aluop)
12345: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12346: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1366345448 (aluop)
12347: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         3 (aluop)
12348: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12349: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1366678054 (aluop)
12350: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         2 (aluop)
12351: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12352: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1367010660 (aluop)
12353: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         1 (aluop)
12354: pwr=1, pc= 26, npc= 24, code=9128, branch to  24
12355: pwr=1, pc= 24, npc= 25, code=2331, mem[ 3]<=1367343266 (aluop)
12356: pwr=1, pc= 25, npc= 26, code=3224, mem[ 2]<=         0 (aluop)
12357: pwr=1, pc= 26, npc= 27, code=9128, branch to  24
12358: pwr=1, pc= 27, npc= 27, code=ffff, stop detected
12359: pc= 27, code=ffff, program stopped
L47 "cpu32_test.v": $finish at simulation time 61797400
0 simulation events (use +profile or +listcounts option to count) + 127969435 accelerated events + 107188926 timing check eve
nts
CPU time: 0.0 secs to compile + 0.1 secs to link + 106.8 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Apr 23, 2018  15:46:02
Elapsed Time = 1:48.32, CPU Time = 67.436
saoki@saturn.ece.iit.edu:~%
```

Terminal
File Edit View Terminal Tabs Help
Terminal
final_otheradder - File Mana...
15:49

Figure 8: Post Place and Route Simulation Result for a 32 Bit CPU with CLA



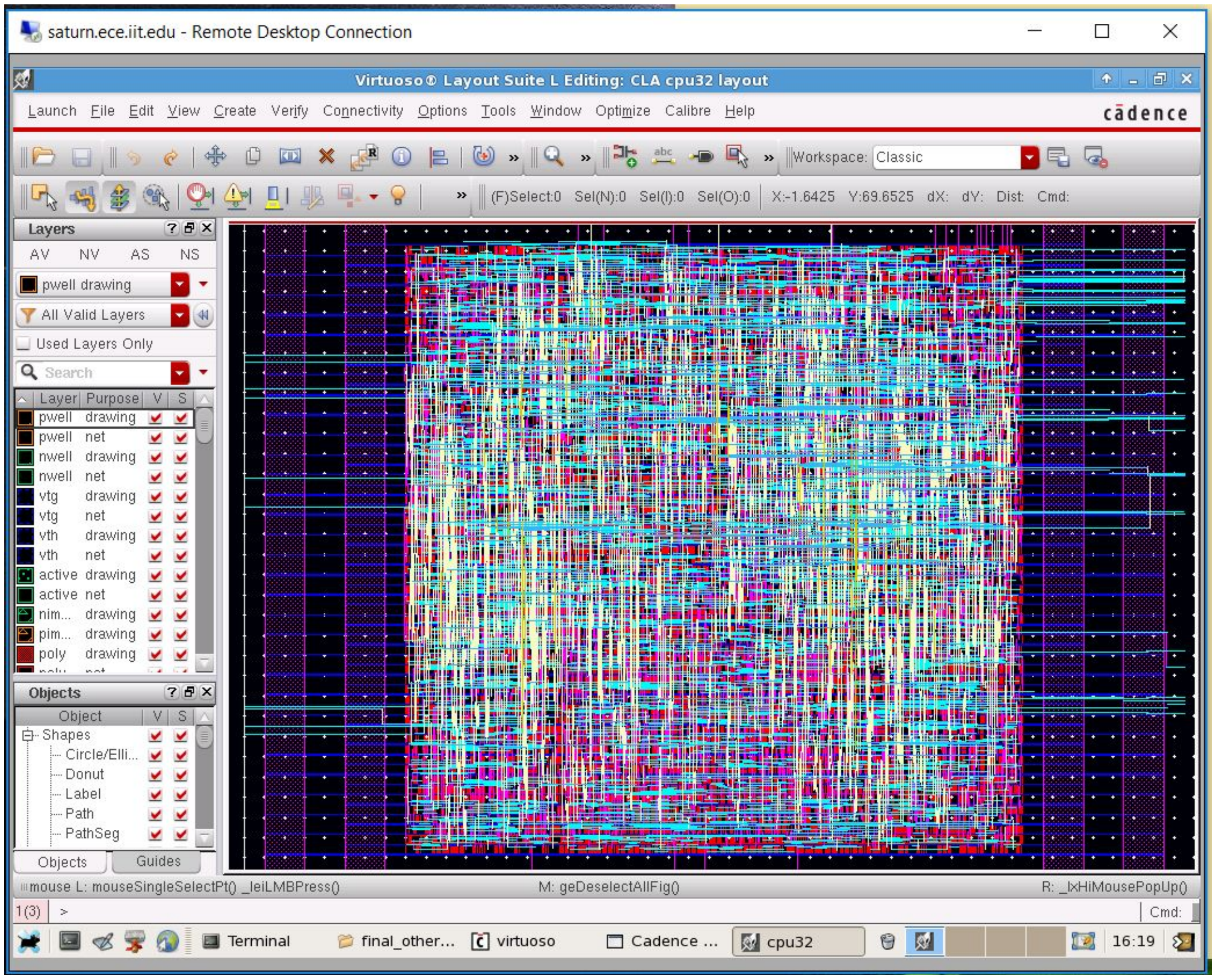
Reference: r:/WORK/cpu32
Implementation: i:/WORK/cpu32

Verification succeeded!

```
Failing (not equivalent)      0       0       0       0       0       0       0       0
***************************************************************************************
1
```

Formality (match)>

Figure 9: Equivalence Checking for 32 Bit CPU with CLA


saturn.ece.iit.edu - Remote Desktop Connection
Virtuoso® Layout Suite L Editing: CLA cpu32 layout
Launch File Edit View Create Verify Connectivity Options Tools Window Optimize Calibre Help
cādence
Workspace: Classic
(F)Select:0 Sel(N):0 Sel(I):0 Sel(O):0 X:-1.6425 Y:69.6525 dX: dY: Dist: Cmd:
Layers
AV NV AS NS
pwell drawing
All Valid Layers
Used Layers Only
Search
Objects
Guides
mouse L: mouseSingleSelectPt() _leiLMBPress()
M: geDeselectAllFig()
R: _lxHiMousePopUp()
Terminal
final_other...
virtuoso
Cadence ...
cpu32
16:19

Layer	Purpose	V	S
pwell	drawing	✓	✓
pwell	net	✓	✓
nwell	drawing	✓	✓
nwell	net	✓	✓
vtg	drawing	✓	✓
vtg	net	✓	✓
vth	drawing	✓	✓
vth	net	✓	✓
active	drawing	✓	✓
active	net	✓	✓
nim...	drawing	✓	✓
pim...	drawing	✓	✓
poly	drawing	✓	✓

Object	V	S
Shapes	✓	✓
Circle/Elli...	✓	✓
Donut	✓	✓
Label	✓	✓
Path	✓	✓
PathSeg	✓	✓

Figure 10: Layout of a 32 Bit CPU with CLA