



# On EE457: Digital IC Design

## Fall Semester 2019

### Final Report Cover Sheet

Due: Monday 3:30PM – 5:45PM, 12/16/2019

**PROJECT TITLE:** 32-Bit Carry Look-Ahead Adder

Group Name: Team DK

Student Names: Kevin Chen (Team Leader) & Darius San Agustin

You Check for completion here	Topics	GRADES
Required	Section1: Executive Summary	
✓	Section 2: Introduction and Background	/5
✓	Section 3: Electric Circuit Schematics	/10
✓	Section 4: Detailed Electric Layouts	/25
✓	Section 5: IRSIM Logic Simulations and Measurements for Layout and Schematic (must provide comparisons between the two)	/10
✓	Section 6: LTSPICE code and parasitic extractions with calculation analysis. Put only samples of code.	/15
✓	Section 7: Measurements in LTSPICE for delays for Layout and Schematic (must provide comparisons between the two)	/15
✓	Section 8: Measurements of power, delay, chip area, timing, number of transistors for the layout. (If you are using TG, static or dynamic CMOS, compare here.)	/10
Required	Section 9: Conclusion and References	
Required	Presentation (Graded by students)	/10
	<b>TOTAL</b>	<b>/100</b>

Instructor Comments:

## Table of Contents

Section 1: Executive Summary .....	3
Section 2: Introduction and Background .....	4
Section 3: Electric Circuit Schematic .....	6
Section 4: Detailed Electric Layout .....	14
Section 5: IRSIM Logic Simulations and Measurements.....	29
Section 5.1: Schematic .....	30
Section 5.2: Layout .....	42
Section 5.3: Comparison .....	46
Section 6: LTSPICE Code and Parasitic Extractions .....	47
Section 6.1: The SPICE Code .....	47
Section 6.2: Parasitic Analysis.....	55
Section 7: Measurements in LTSPICE for Delays .....	66
Section 7.1: Schematic .....	66
Section 7.2: Layout .....	70
Section 7.3: Comparison .....	72
Section 8: Measurement of Power, Chip Area, and Transistor Count.....	73
Section 8.1: Power Measurement .....	73
Section 8.2: Total Chip Area .....	73
Section 8.3: Total Transistor Count .....	74
Section 9: Conclusion .....	75
References.....	76

## Section 1: Executive Summary

For this final project, we design and build a 32-bit Carry Look-Ahead adder. We use old components designed in Project 2—the inverter, NAND gate, XOR gate, and full adder—to realize this design. To do this, we first modify the full adder to output two signals, propagate ( $P$ ) and generate ( $G$ ), as opposed to the carry-out signal. These two signals are used in the Look-Ahead Carry Unit (LCU) modules that will compute the carry-in signals for all full adders in constant time. The constant-time computation of carry-in signals, as opposed to the rippling of them in the Ripple-Carry Adder design, is the essence of the Carry Look-Ahead Adder.

We then design C1, C2, and C3 modules that compute the carry-in signals for the latter three full adders. The carry-in signal for the first full adder is inputted directly from outside the circuit. This will realize a four-bit CLA design. But to scale up, we use another C3 module and a four-input AND gate (called the PG4 module) to output propagate and generate signals for the entire four-bit CLA.

We then use the four-bit CLA to build a 16-bit CLA, using four four-bit CLAs and the same LCU modules. We then use the 16-bit CLA to build up once again to the final circuit, the 32-bit Carry Look-Ahead Adder. As in previous projects, we verify the functionality of the design by performing numerical verifications—five additions and five two's complement subtractions—in IRSIM. We also measure the circuit's timing and other metrics using LTSPICE. Specifically, we measure the output rise time, fall time, and propagation delays for both schematic and layout, then compare their values. We also measure the total chip area, the total transistor count of the entire circuit, and the power dissipation of the circuit while it is in steady state (i.e. not switching).

## Section 2: Introduction and Background

In this project, we will build a 32-bit Carry Look-Ahead adder. The purpose of using Carry Look-Ahead logic is to enable the circuit to calculate carry-input signals for all full adders at the same time, instead of waiting for the carry-out signal from one FA to ripple into the next. Because all carry signals are computed at the same time, this enables this circuit to perform binary addition faster than the Ripple-Carry Adder. Here, we see a phenomenon known as the Space-Time tradeoff, where we increase the physical space used by the circuit in order to decrease the time it takes for the circuit to perform its intended function [1].

To accomplish this design, we modify some of the circuits that we developed in the making of the 16-bit ripple carry adder. The main modification was made to the full adder. The sum bit is still computed, but since the carry-out of one FA no longer ripples into the next, that output was removed. Instead, the propagate P and generate G signals were computed. Both of these signals are used in the computation of carry-in signals for all full adders in an adder unit. Those signals have the below Boolean functions, where the subscript i refers to a specific full adder, where  $i = 0$  for the first full adder [2].

$$P_i = A_i \oplus B_i = \bar{A}_i B_i + A_i \bar{B}_i$$

$$G_i = A_i \cdot B_i$$

These signals will be used by the Look-Ahead Carry Unit, or LCU, to determine the carry-in values for all full adders at the same time. The LCU is the bread and butter of this project, as it adds circuit complexity to perform binary additions faster. Given the carry input  $C_{in}$  into the first full adder, the below equations are used by the LCU to calculate the carry inputs for the other three full adders to build a 4-bit Carry Look-Ahead Adder [2].

$$C_1 = G_0 + P_0 C_{in}$$

$$C_2 = G_1 + P_1 G_0 + P_1 P_0 C_{in}$$

$$C_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in}$$

$$C_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{in}$$

Note that  $C_4$  is the carry-out signal of the entire four bit adder. It's not necessary to perform binary addition, so we did not implement it. Only the  $C_1$ ,  $C_2$ , and  $C_3$  modules were implemented using two-input NAND gates and inverters from previous projects.

On top of those Boolean functions, we also implement and output the group propagate and group generate signals to allow us to scale up with these 4-bit adders [3].

$$P_G = P_3 P_2 P_1 P_0$$

$$G_G = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0$$

Note that  $G_G$  is the same form as C3! We can easily implement  $G_G$  by reusing the module for C3. For  $P_G$ , we implement a simple four input AND gate using static CMOS design.

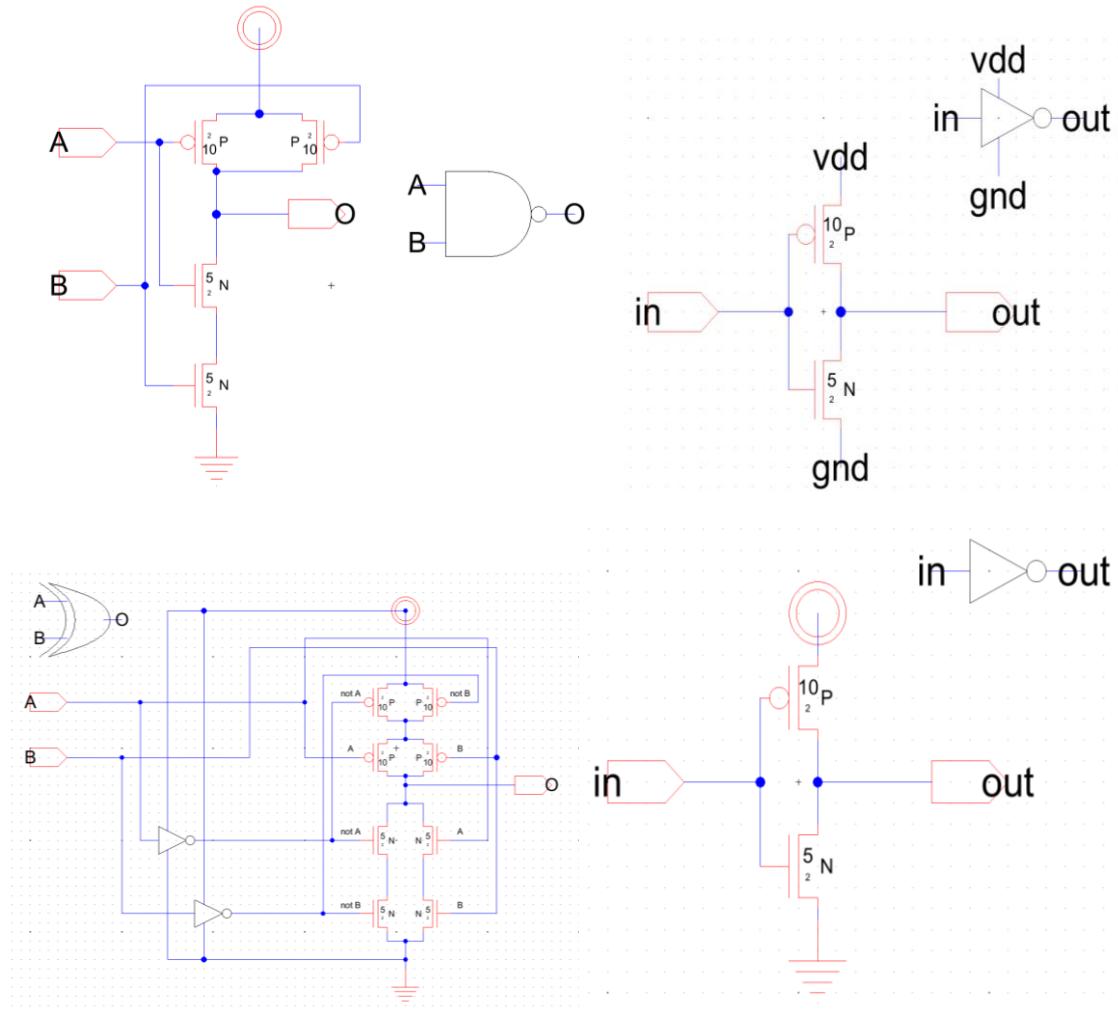
Now that our four-bit CLA module is complete, we can now use it to build a 16-bit CLA. The same LCU modules are used here, and they take in the  $P_G$  and  $G_G$  outputs of each 4-bit CLA. And just as we calculated  $P_G$  and  $G_G$  for the four-bit adders, we also calculate it for the 16-bit adder.

With our 16-bit CLA module finished, we scale up once more to at 32-bit CLA. Because only two 16-bit CLA modules are used, only the C1 LCU module is required. Because this is the final stage, the Cin input is grounded, and  $G_G$  and  $P_G$  are not calculated. Only the  $G_G$  and  $P_G$  output of the first 16-bit CLA is used (for calculation of C1). The  $G_G$  and  $P_G$  outputs of the second 16-bit CLA are grounded, just like Cin. No carry-out bit is computed because detecting arithmetic overflow is out of the scope of this project. Therefore, the only inputs are the 32-bit A and B inputs. The only outputs are the 32 sum bits.

For this project, we have selected  $W_N = 5\lambda$  and  $W_P = 10\lambda$ . We also keep the technology scale  $\lambda = 175$  nm.

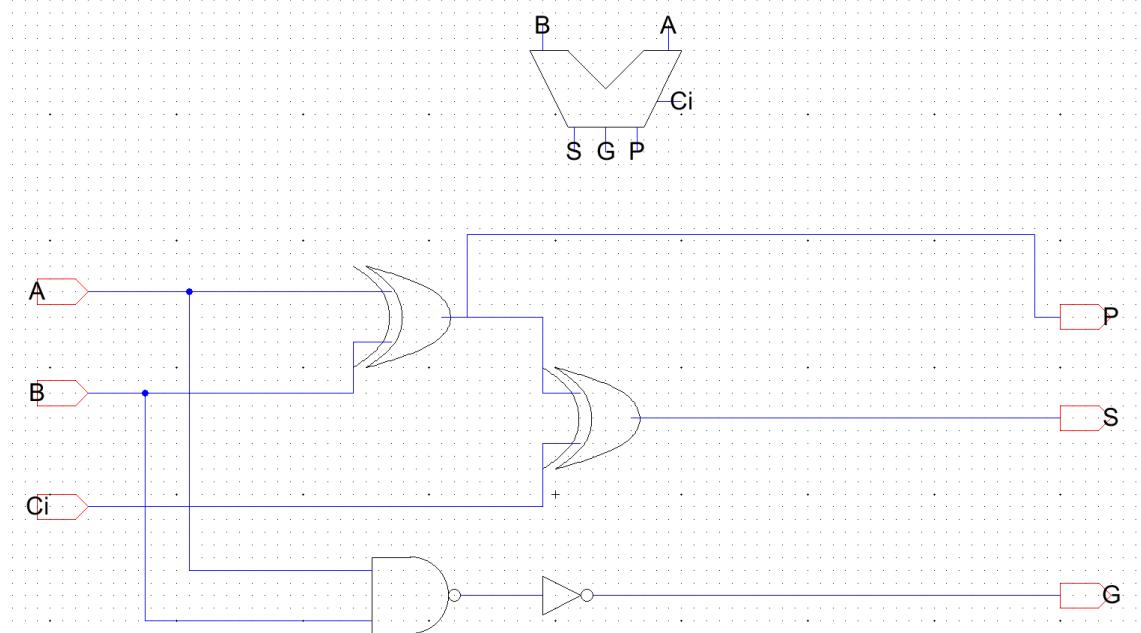
### Section 3: Electric Circuit Schematic

In this section, we discuss the schematic of our 32-bit Carry Look-Ahead Adder, starting with its sub-components. Below are the NAND, inverter, and XOR schematics seen in previous projects.



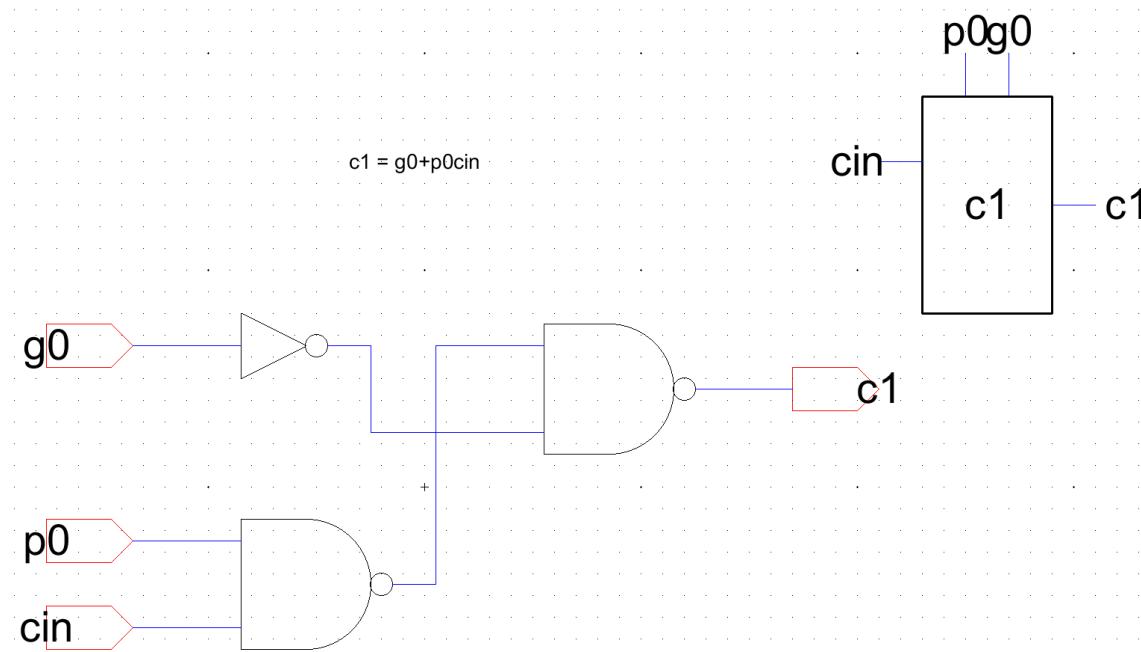
**Figure 1: Old Schematics - NAND (Top Left), Inverter without Wells (Top Right), XOR Gate (Bottom Left), Inverter with Wells (Bottom Right)**

Below is the modified full adder that outputs the sum bit S, the propagate bit P, and the generate bit G.



**Figure 2: Modified Full Adder Schematic**

The modules used in creating the LCU are shown below. In order, they are the C1, C2, C3, and PG4 modules. Note that the PG4 module is simply a four-input static CMOS AND gate.



**Figure 3: C1 Module Schematic**

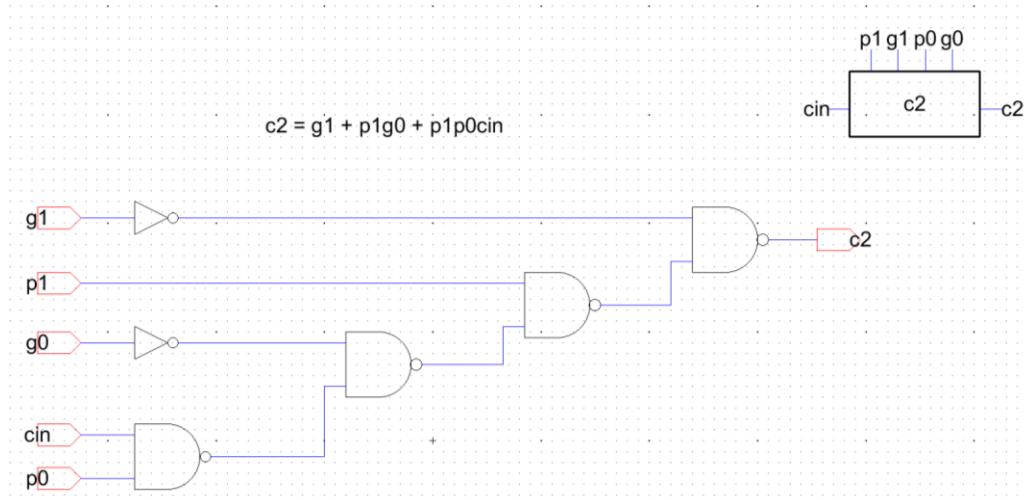


Figure 4: C2 Module Schematic

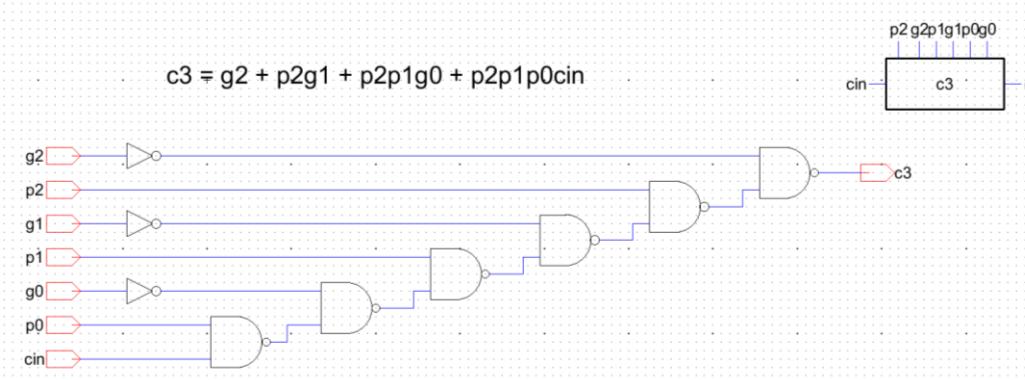


Figure 5: C3 Module Schematic

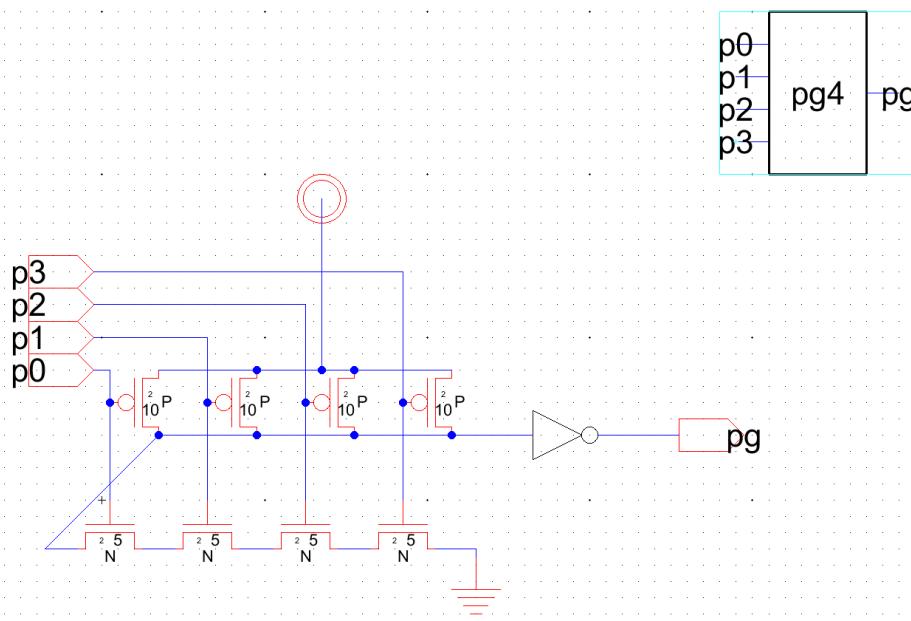
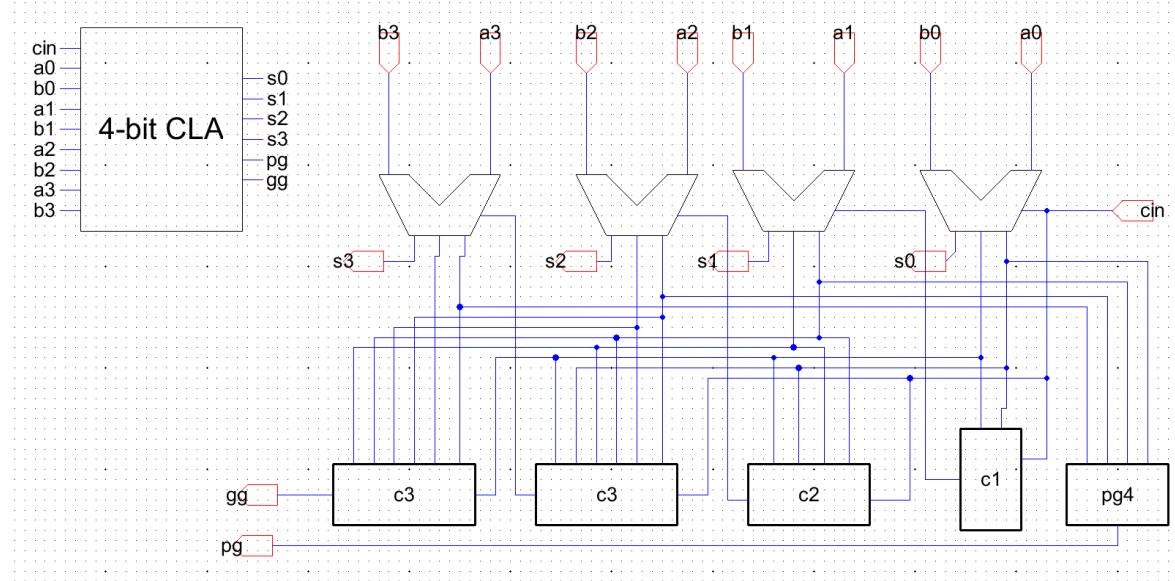


Figure 6: PG4 Module Schematic

Finally, four of these full adders, together with these LCU modules are used to create the four-bit carry look-ahead adder below.



**Figure 7: 4-bit CLA Schematic**

We then use the 4-bit CLA to scale up to a 16-bit adder, using the same LCU modules to make it possible. On the next page is the schematic.

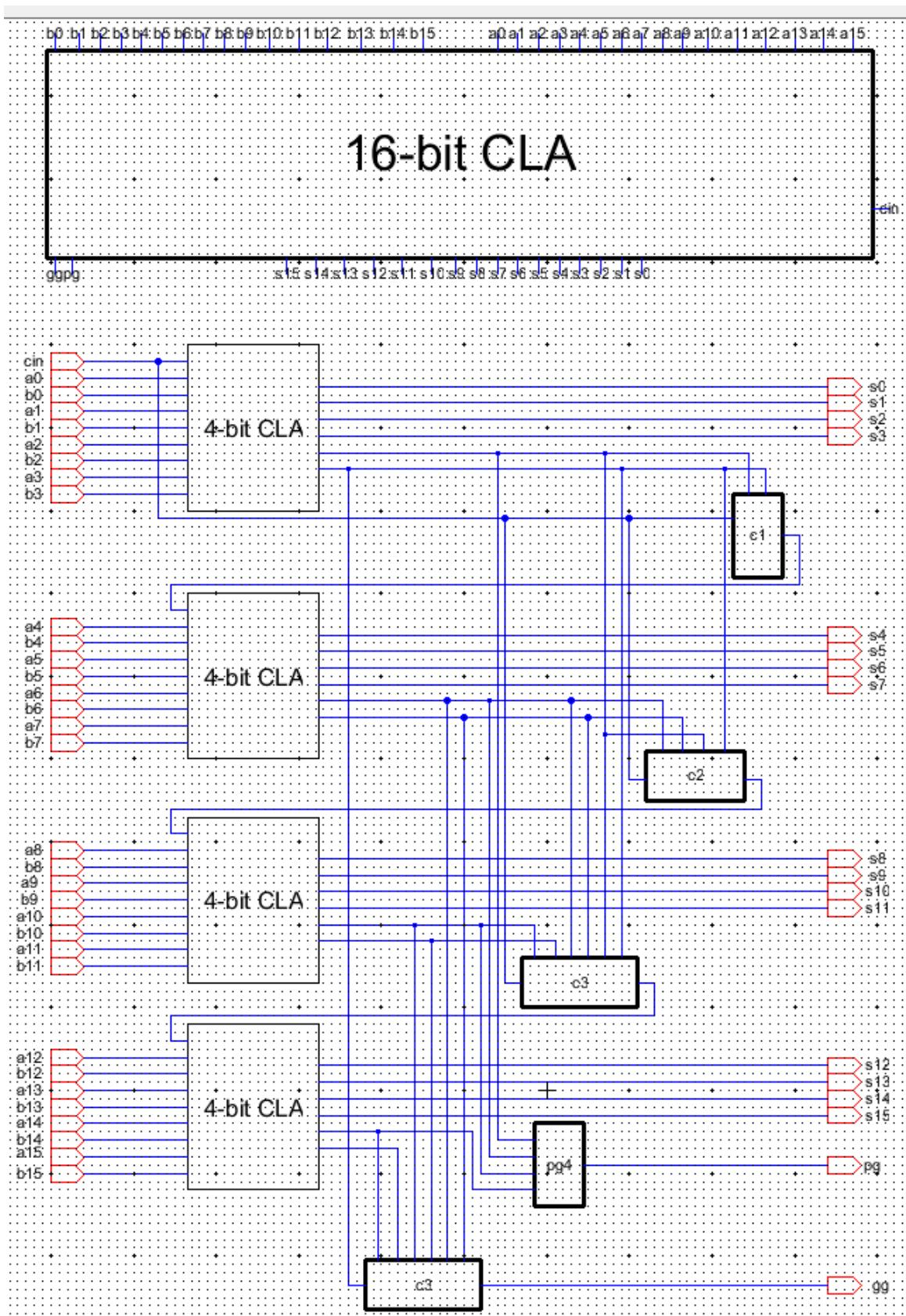


Figure 8: 16-bit CLA Schematic

We scale up again to the final 32-bit adder. This time, we only need to use the C1 module to realize this design. Below is the schematic.

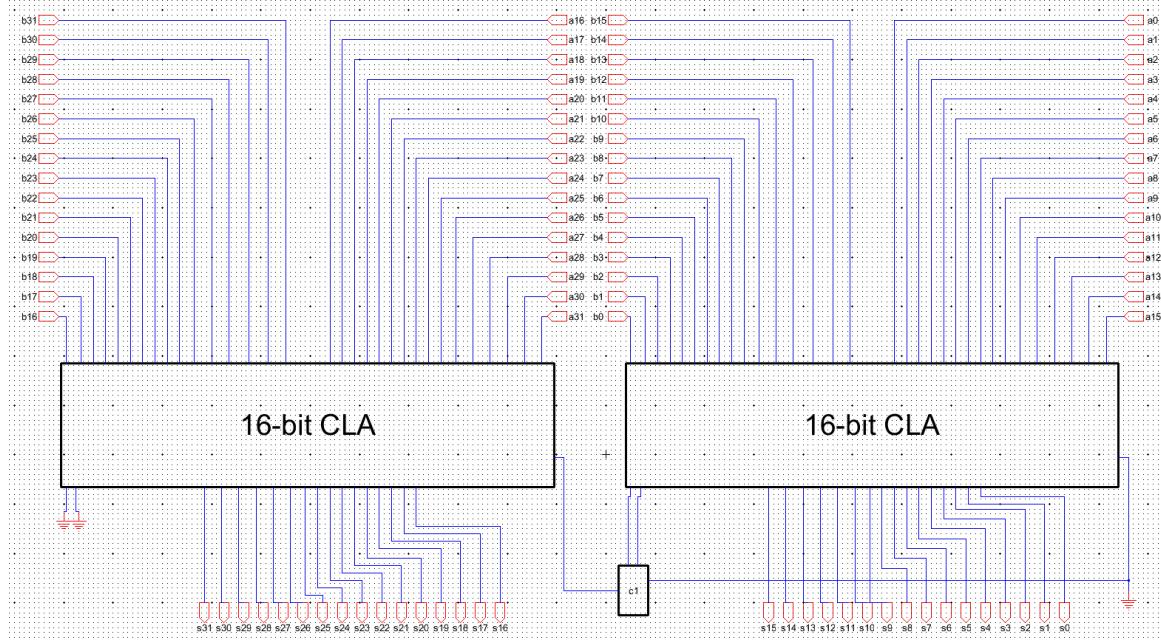


Figure 9: 32-bit CLA Schematic

On the next page is the transistor-only schematic of the 32-bit CLA, which does not use icons.

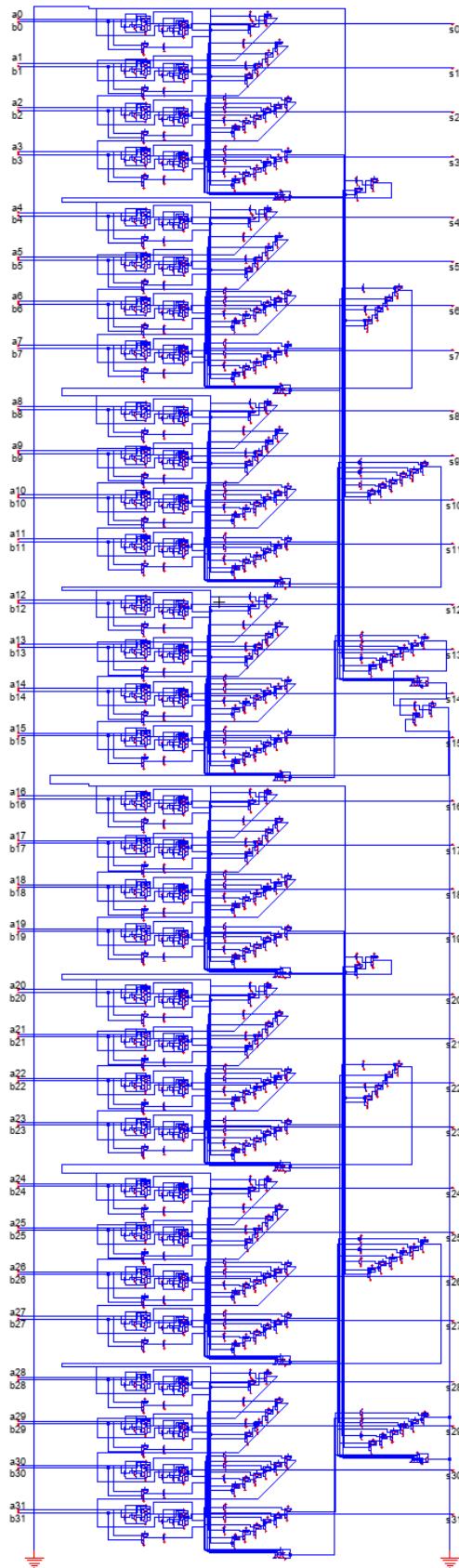
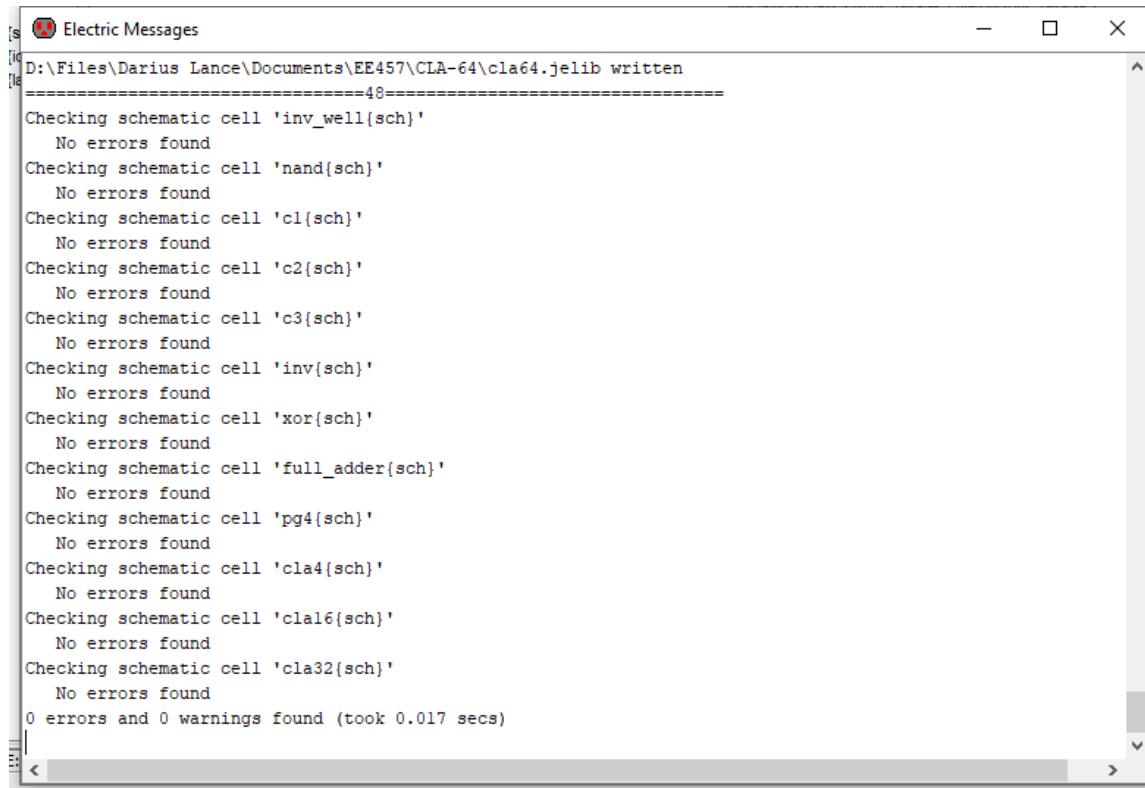


Figure 10: 32-bit CLA Schematic (Only Transistors)

Below are the schematic DRC checks of the highest level circuit (32-bit CLA). This DRC also performs the DRC checks for all the lower-level components. No errors are found.



```
[s] Electric Messages
[D:\Files\Darius Lance\Documents\EE457\CLA-64\cla64.jelib written
=====
=====48=====
Checking schematic cell 'inv_well{sch}'
  No errors found
Checking schematic cell 'nand{sch}'
  No errors found
Checking schematic cell 'cl1{sch}'
  No errors found
Checking schematic cell 'c2{sch}'
  No errors found
Checking schematic cell 'c3{sch}'
  No errors found
Checking schematic cell 'inv{sch}'
  No errors found
Checking schematic cell 'xor{sch}'
  No errors found
Checking schematic cell 'full_adder{sch}'
  No errors found
Checking schematic cell 'pg4{sch}'
  No errors found
Checking schematic cell 'cla4{sch}'
  No errors found
Checking schematic cell 'cla16{sch}'
  No errors found
Checking schematic cell 'cla32{sch}'
  No errors found
0 errors and 0 warnings found (took 0.017 secs)
```

Figure 11: 32-bit CLA Schematic DRC

#### Section 4: Detailed Electric Layout

For reference, the layouts of old components—the NAND gate, inverter, and XOR gate—are included below. Their checks will not be included, because their correctness has already been verified in previous projects.

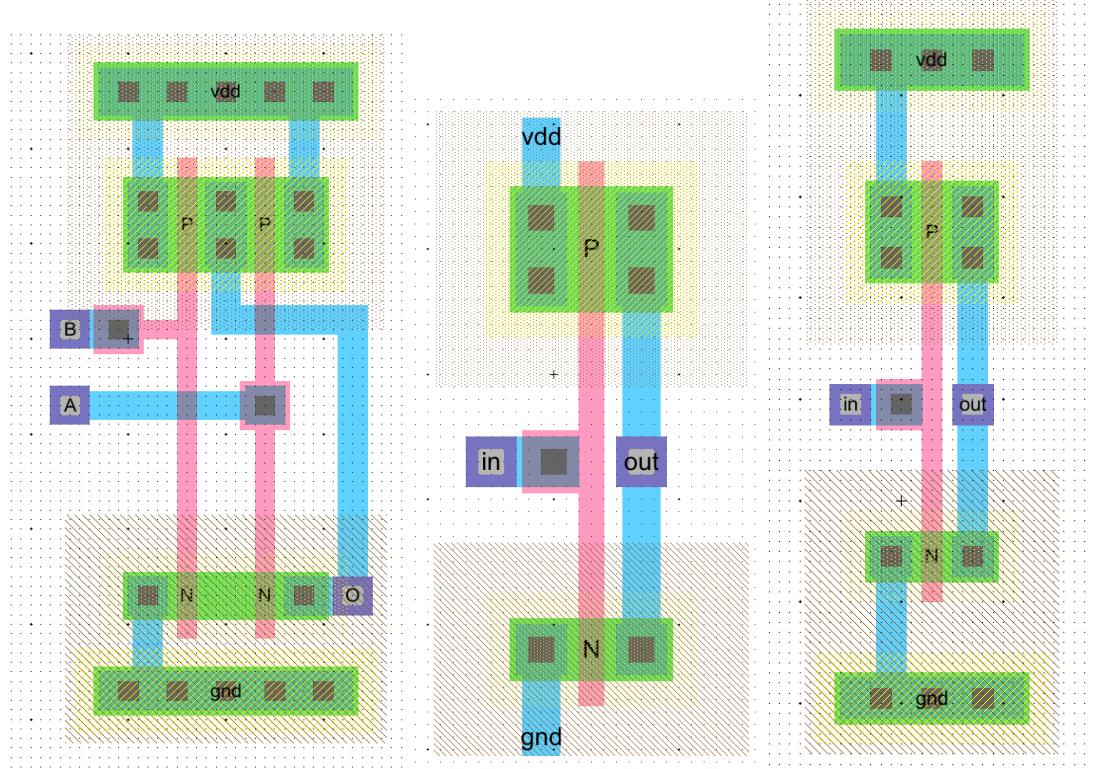


Figure 12: Layouts of Old Components - NAND (Left), Well-less Inverter (Middle), Inverter with Well (Right)

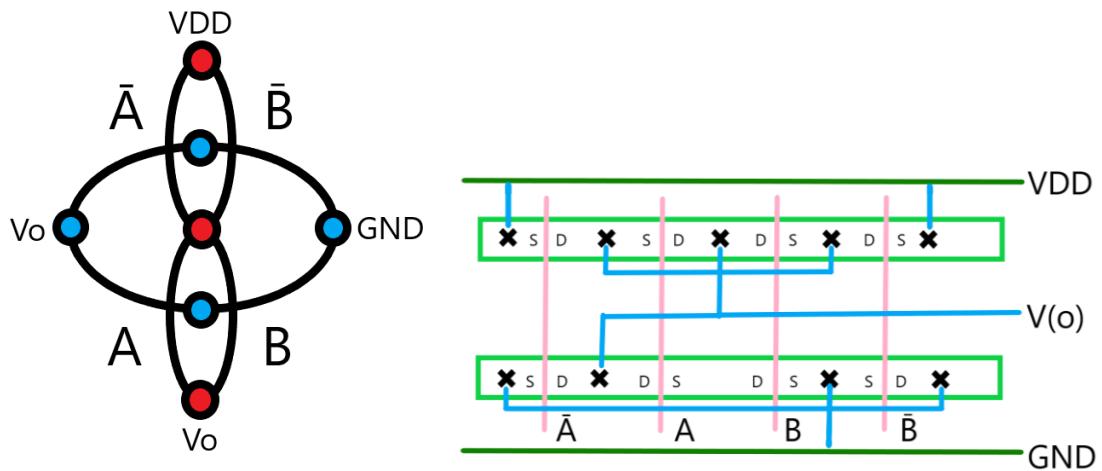
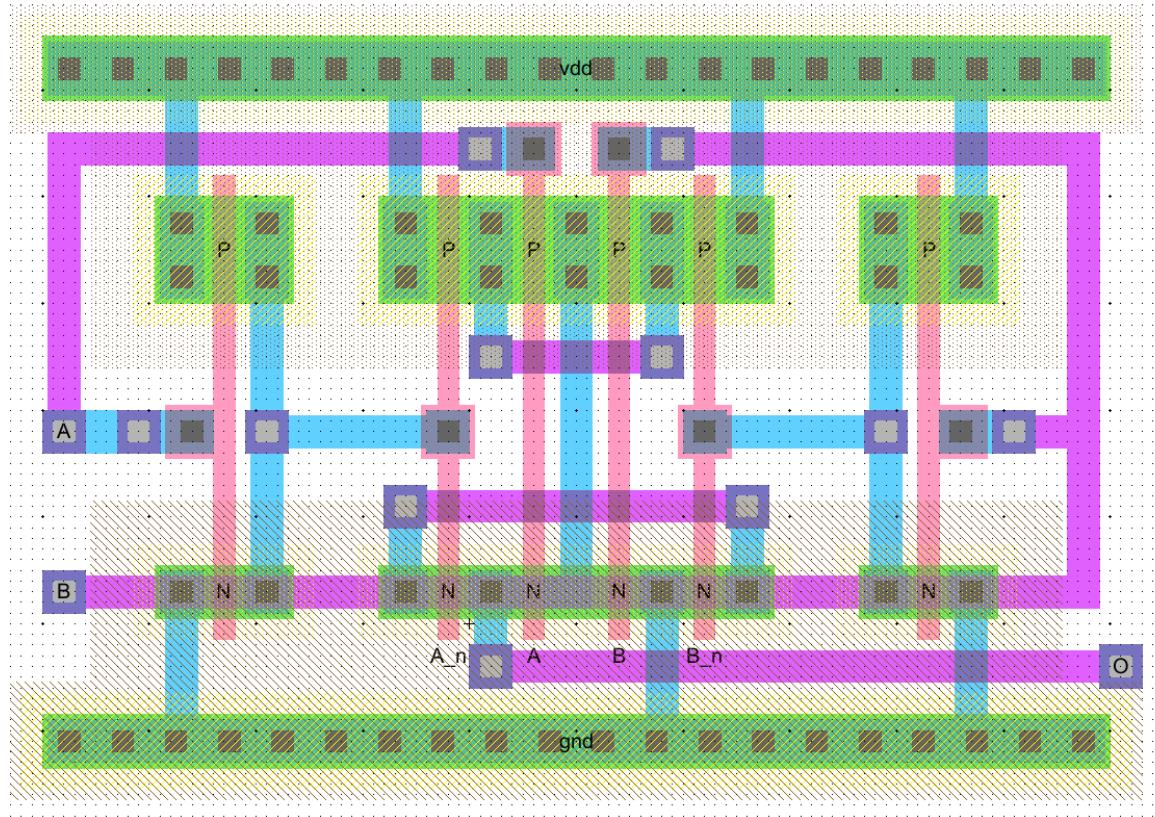
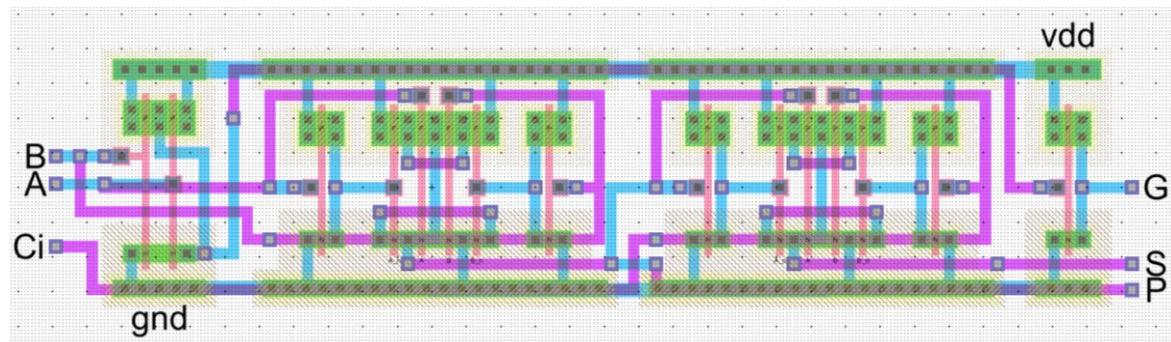


Figure 13: Euler Graph (Right) and Stick Diagram (Left) of XOR Gate



**Figure 14: Layout of XOR Gate**

Below is the layout of the modified full adder. It is followed by the C1, C2, C3, and PG4 LCU modules. Each layout is accompanied by its respective DRC, ERC Well Check, and NCC output from the Electric logs. This is to verify that each component follows its respective schematic and all design rules.



**Figure 15: Modified Full Adder Layout**

```

Electric Messages
round 1v networks
0 errors and 0 warnings found (took 0.003 secs)
=====
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.0 secs)
Found 10 networks
0 errors and 0 warnings found (took 0.001 secs)
=====
Checking Wells and Substrates in 'cla64:full_adder{lay}' ...
    Geometry collection found 84 well pieces, took 0.001 secs
    Geometry analysis used 4 threads and took 0.002 secs
NetValues propagation took 0.0 secs
Checking short circuits in 8 well contacts
    Additional analysis took 0.0 secs
No Well errors found (took 0.003 secs)
=====
Hierarchical NCC every cell in the design: cell 'full_adder{sch}' cell 'full_adder{lay}'
Comparing: cla64:inv_well{sch} with: cla64:inv_well{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:nand{sch} with: cla64:nand{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:inv{sch} with: cla64:inv{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:xor{sch} with: cla64:xor{lay}
    exports match, topologies match, sizes match in 0.002 seconds.
Comparing: cla64:full_adder{sch} with: cla64:full_adder{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Summary for all cells: exports match, topologies match, sizes match
NCC command completed in: 0.008 seconds.

```

Figure 16: Full Adder Checks

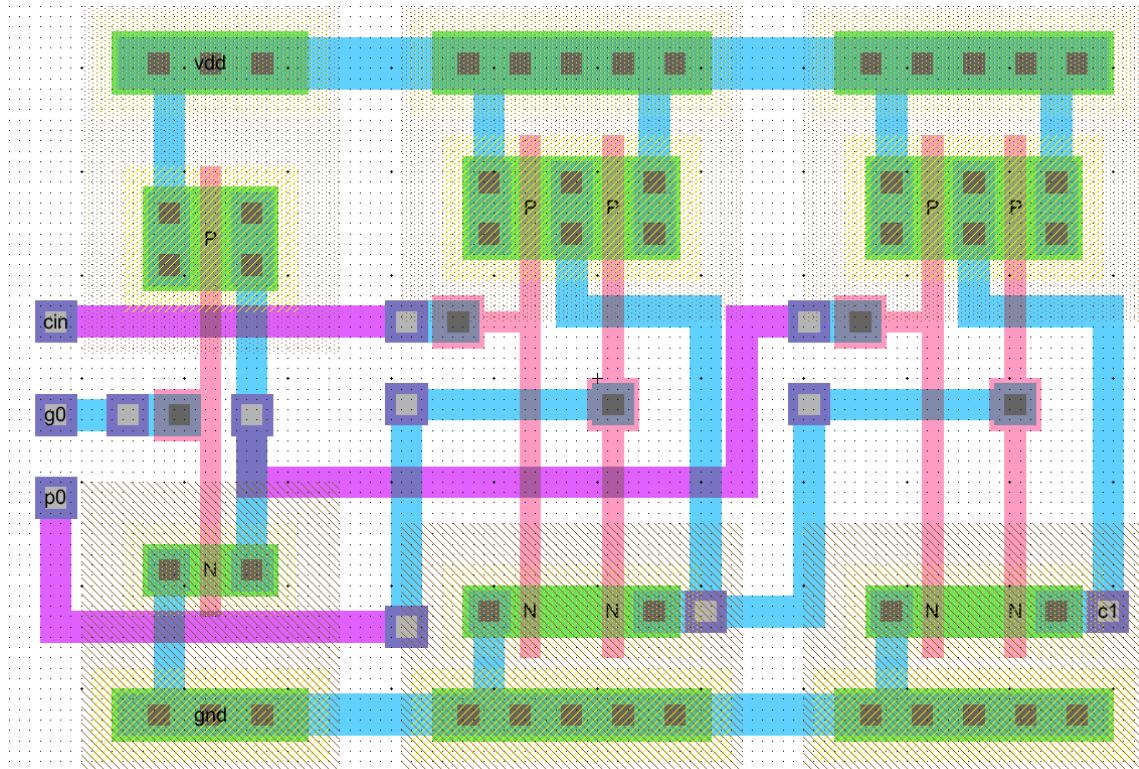
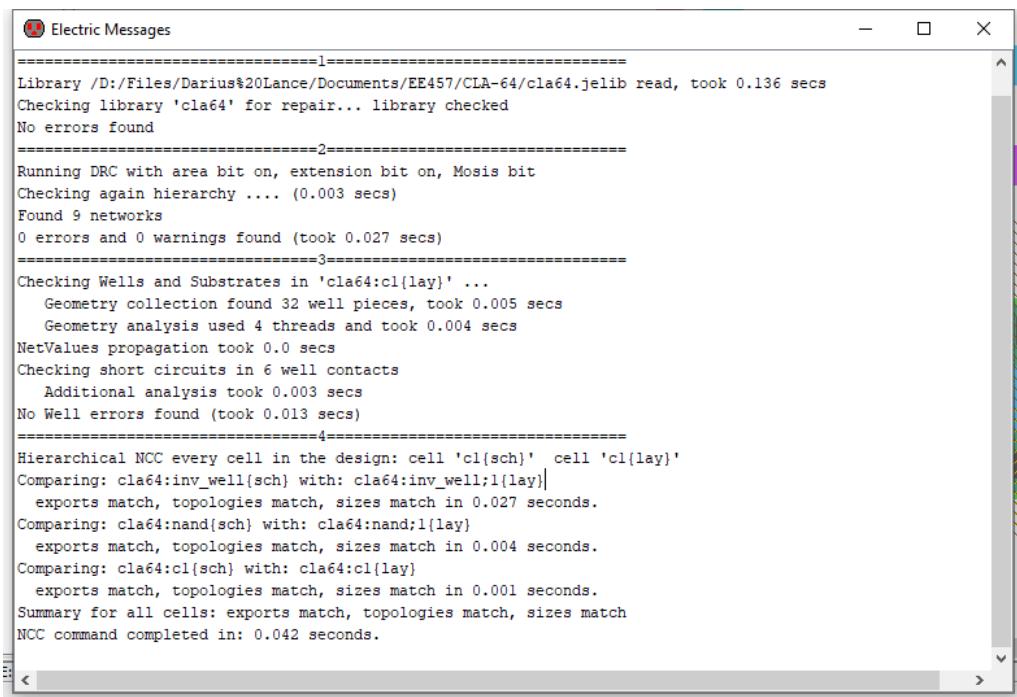


Figure 17: C1 Module Layout



```

====1=====
Library /D:/Files/Darius%20Lance/Documents/EE457/CLA-64/cla64.jelib read, took 0.136 secs
Checking library 'cla64' for repair... library checked
No errors found
====2=====
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.003 secs)
Found 9 networks
0 errors and 0 warnings found (took 0.027 secs)
====3=====
Checking Wells and Substrates in 'cla64:cl(lay)' ...
    Geometry collection found 32 well pieces, took 0.005 secs
    Geometry analysis used 4 threads and took 0.004 secs
NetValues propagation took 0.0 secs
Checking short circuits in 6 well contacts
    Additional analysis took 0.003 secs
No Well errors found (took 0.013 secs)
====4=====
Hierarchical NCC every cell in the design: cell 'cl(sch)' cell 'cl(lay)'
Comparing: cla64:inv_well(sch) with: cla64:inv_well;l(lay)
    exports match, topologies match, sizes match in 0.027 seconds.
Comparing: cla64:nand(sch) with: cla64:nand;l(lay)
    exports match, topologies match, sizes match in 0.004 seconds.
Comparing: cla64:cl(sch) with: cla64:cl(lay)
    exports match, topologies match, sizes match in 0.001 seconds.
Summary for all cells: exports match, topologies match, sizes match
NCC command completed in: 0.042 seconds.

```

Figure 18: C1 Module Checks

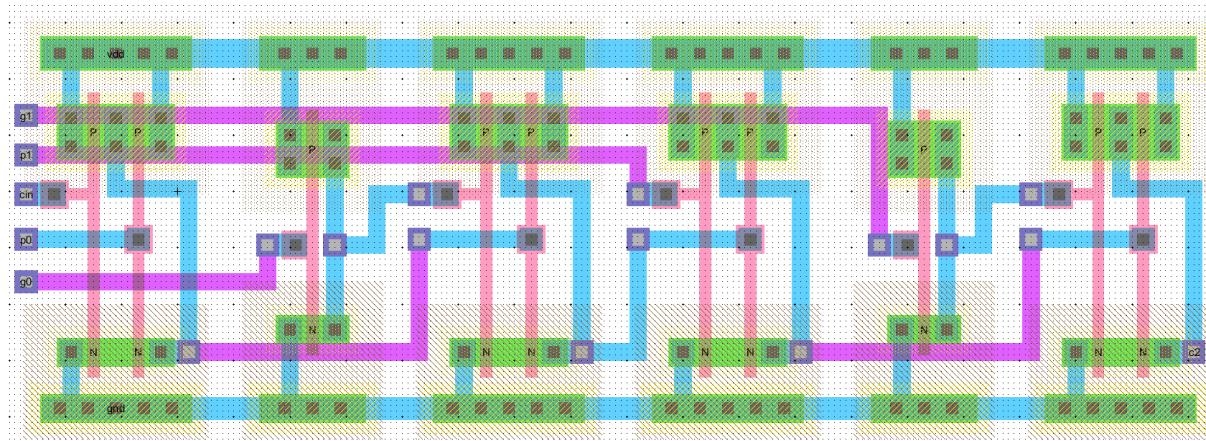


Figure 19: C2 Module Layout

```

Electric Messages
D:\Files\Darius Lance\Documents\EE457\CLA-64\cla64.jelib written
=====
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.0 secs)
Found 14 networks
0 errors and 0 warnings found (took 0.001 secs)
=====
31=====
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.0 secs)
Found 14 networks
0 errors and 0 warnings found (took 0.001 secs)
=====
32=====
Checking Wells and Substrates in 'cla64:c2{lay}' ...
    Geometry collection found 64 well pieces, took 0.0 secs
    Geometry analysis used 4 threads and took 0.002 secs
NetValues propagation took 0.0 secs
Checking short circuits in 12 well contacts
    Additional analysis took 0.0 secs
No Well errors found (took 0.003 secs)
=====
33=====
Hierarchical NCC every cell in the design: cell 'c2{sch}' cell 'c2{lay}'
Comparing: cla64:inv_well{sch} with: cla64:inv_well;1{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:nand{sch} with: cla64:nand;1{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:c2{sch} with: cla64:c2{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Summary for all cells: exports match, topologies match, sizes match
NCC command completed in: 0.003 seconds.

```

Figure 20: C2 Module Checks

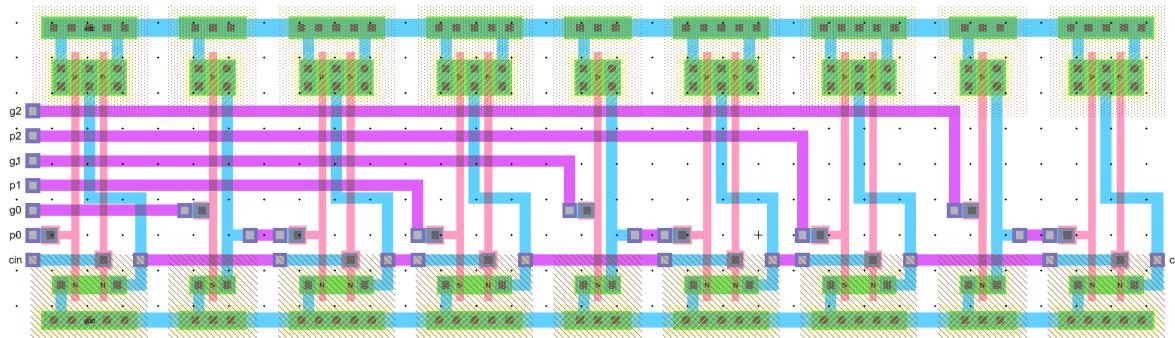


Figure 21: C3 Module Layout

```

Electric Messages
NCC command completed in: 0.007 seconds.
=====
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.0 secs)
Found 19 networks
Checking cell 'c3{lay}'
    No errors/warnings found
0 errors and 0 warnings found (took 0.088 secs)
=====
Checking Wells and Substrates in 'cla64:c3{lay}' ...
    Geometry collection found 96 well pieces, took 0.001 secs
    Geometry analysis used 4 threads and took 0.001 secs
NetValues propagation took 0.001 secs
Checking short circuits in 18 well contacts
    Additional analysis took 0.0 secs
No Well errors found (took 0.004 secs)
=====
Hierarchical NCC every cell in the design: cell 'c3{sch}'  cell 'c3{lay}'

Comparing: cla64:inv_well{sch} with: cla64:inv_well{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:nand{sch} with: cla64:nand{lay}
    exports match, topologies match, sizes match in 0.002 seconds.
Comparing: cla64:c3{sch} with: cla64:c3{lay}
    exports match, topologies match, sizes match in 0.002 seconds.
Summary for all cells: exports match, topologies match, sizes match
NCC command completed in: 0.008 seconds.

```

Figure 22: C3 Module Checks

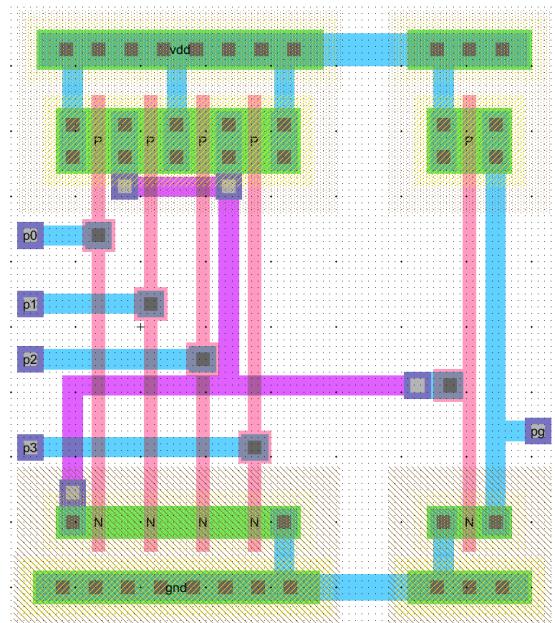
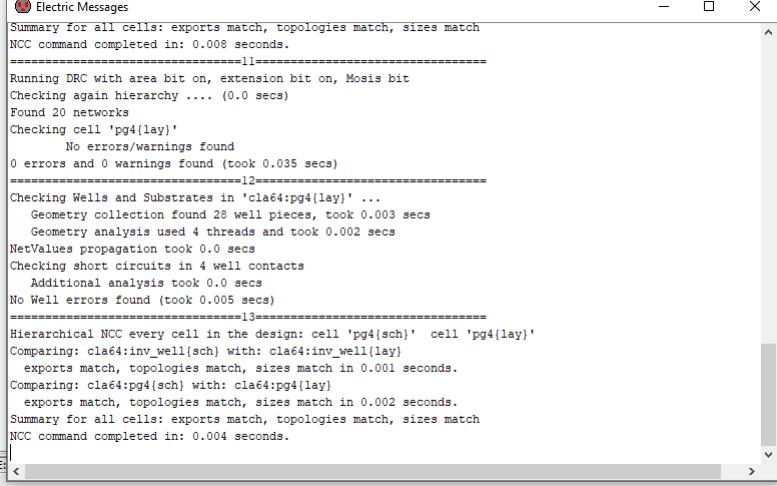


Figure 23: PG4 Module Layout



```

Electric Messages
Summary for all cells: exports match, topologies match, sizes match
NCC command completed in: 0.008 seconds.
=====
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.0 secs)
Found 20 networks
Checking cell 'pg4(lay)'
    No errors/warnings found
0 errors and 0 warnings found (took 0.035 secs)
=====
Checking Wells and Substrates in 'cla4:pg4(lay)' ...
    Geometry collection found 28 well pieces, took 0.003 secs
    Geometry analysis used 4 threads and took 0.002 secs
NetValues propagation took 0.0 secs
Checking short circuits in 4 well contacts
    Additional analysis took 0.0 secs
No Well errors found (took 0.005 secs)
=====
Hierarchical NCC every cell in the design: cell 'pg4(sch)' cell 'pg4(lay)'
Comparing: cla64:inv_well(sch) with: cla64:inv_well(lay)
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:pg4(sch) with: cla64:pg4(lay)
    exports match, topologies match, sizes match in 0.002 seconds.
Summary for all cells: exports match, topologies match, sizes match
NCC command completed in: 0.004 seconds.

```

**Figure 24: PG4 Module Checks**

Note that the C3 module and PG4 module layouts use slightly different versions of the NAND and inverter layout. These alternate versions are slightly taller than their counterparts, creating more space between the pull-up network and the pull-down network. This was done to reduce wiring complexity and to allocate more room for connections to inputs. Below is the layout of the 4-bit CLA.

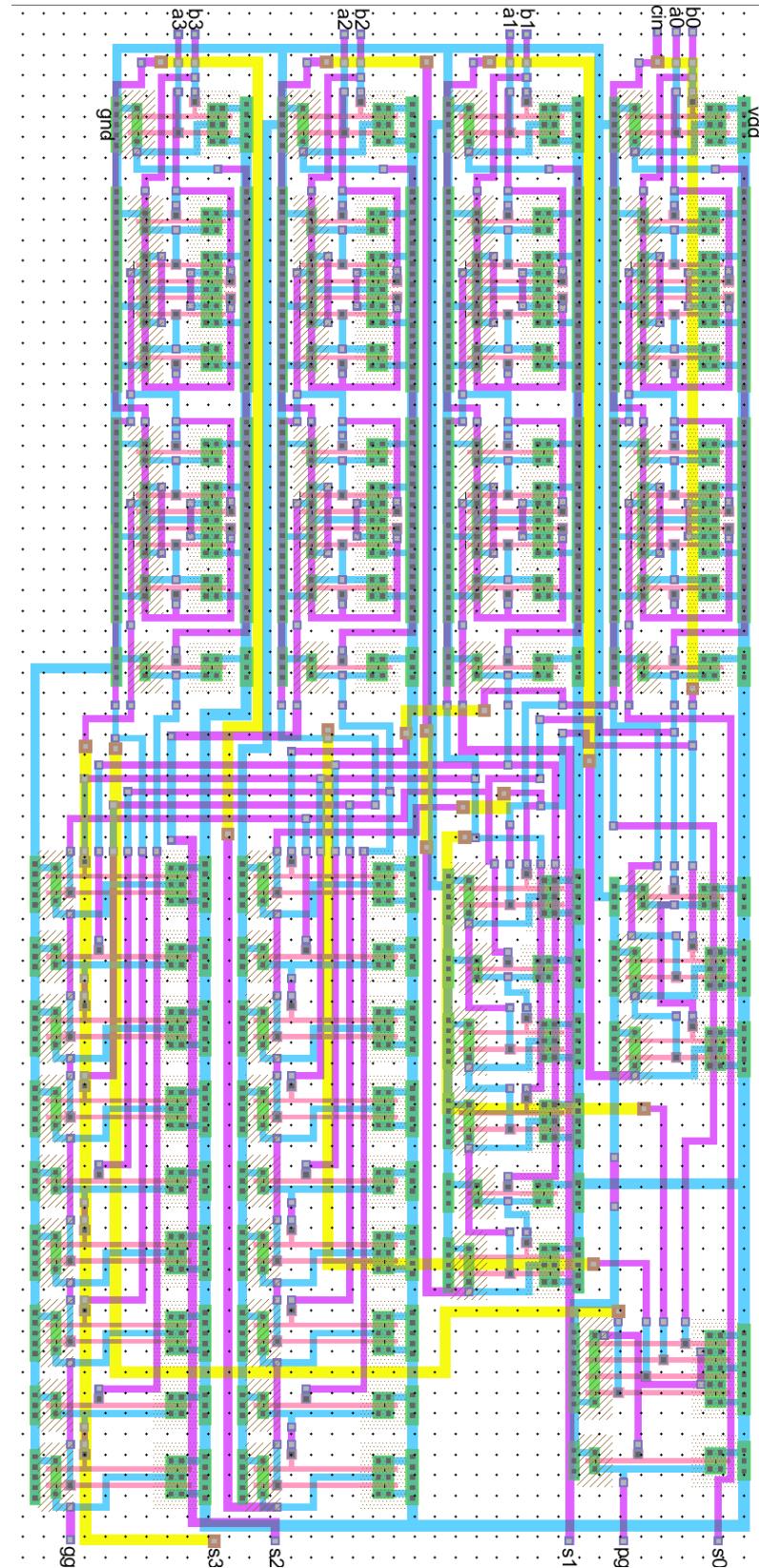
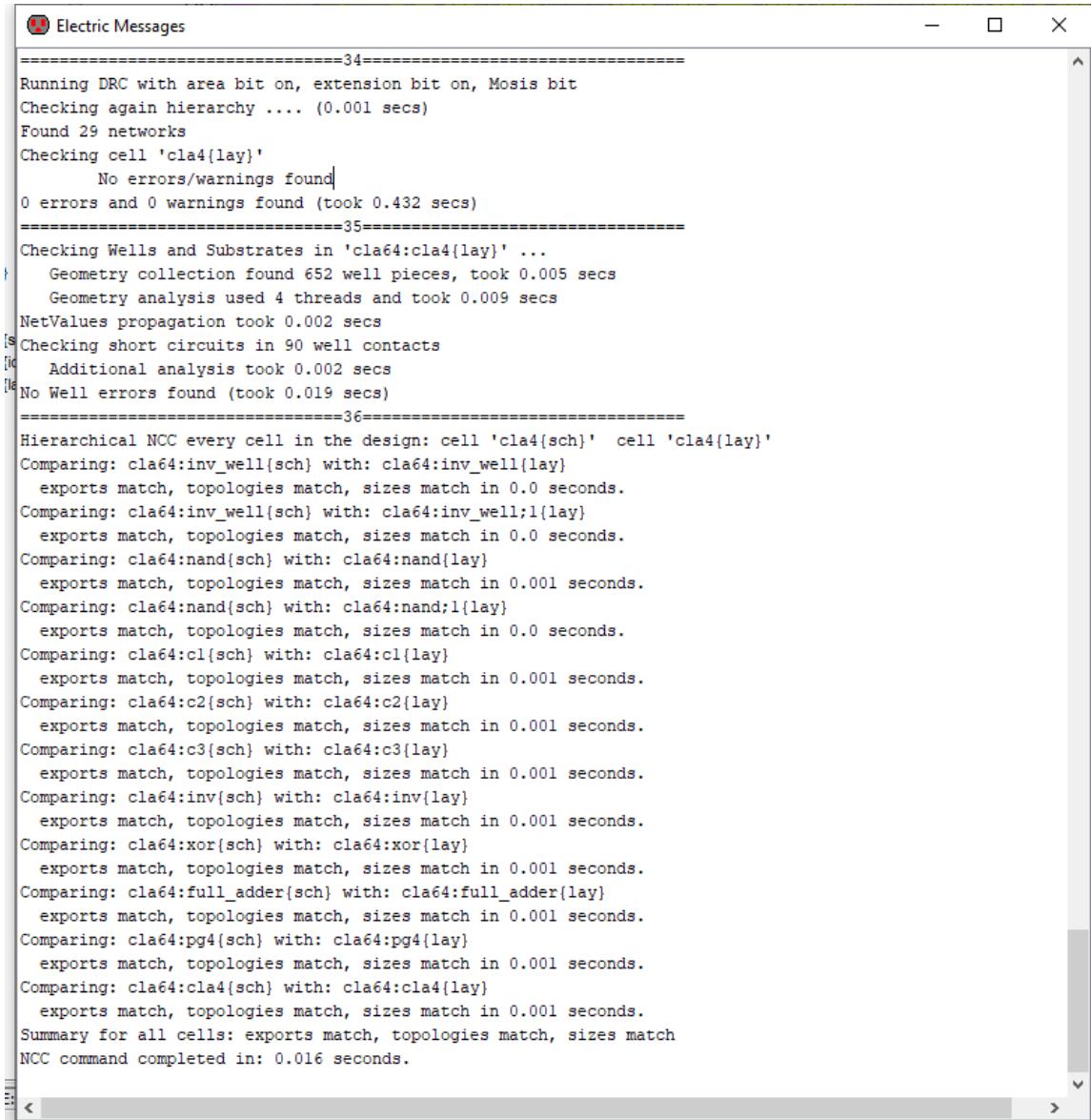


Figure 25: 4-bit CLA Layout (Rotated to show more room)



```

====34=====
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.001 secs)
Found 29 networks
Checking cell 'cla4{lay}'
    No errors/warnings found
0 errors and 0 warnings found (took 0.432 secs)
=====35=====
Checking Wells and Substrates in 'cla64:cla4{lay}' ...
    Geometry collection found 652 well pieces, took 0.005 secs
    Geometry analysis used 4 threads and took 0.009 secs
NetValues propagation took 0.002 secs
[s] Checking short circuits in 90 well contacts
[!] Additional analysis took 0.002 secs
[!] No Well errors found (took 0.019 secs)
=====36=====
Hierarchical NCC every cell in the design: cell 'cla4{sch}' cell 'cla4{lay}'
Comparing: cla64:inv_well{sch} with: cla64:inv_well{lay}
    exports match, topologies match, sizes match in 0.0 seconds.
Comparing: cla64:inv_well{sch} with: cla64:inv_well;l{lay}
    exports match, topologies match, sizes match in 0.0 seconds.
Comparing: cla64:nand{sch} with: cla64:nand{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:nand{sch} with: cla64:nand;l{lay}
    exports match, topologies match, sizes match in 0.0 seconds.
Comparing: cla64:c1{sch} with: cla64:c1{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:c2{sch} with: cla64:c2{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:c3{sch} with: cla64:c3{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:inv{sch} with: cla64:inv{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:xor{sch} with: cla64:xor{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:full_adder{sch} with: cla64:full_adder{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:pg4{sch} with: cla64:pg4{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:cla4{sch} with: cla64:cla4{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Summary for all cells: exports match, topologies match, sizes match
NCC command completed in: 0.016 seconds.

```

**Figure 26: 4-bit CLA Checks**

On the next page is the layout of the 16-bit CLA, along with a screenshot of all of its checks.

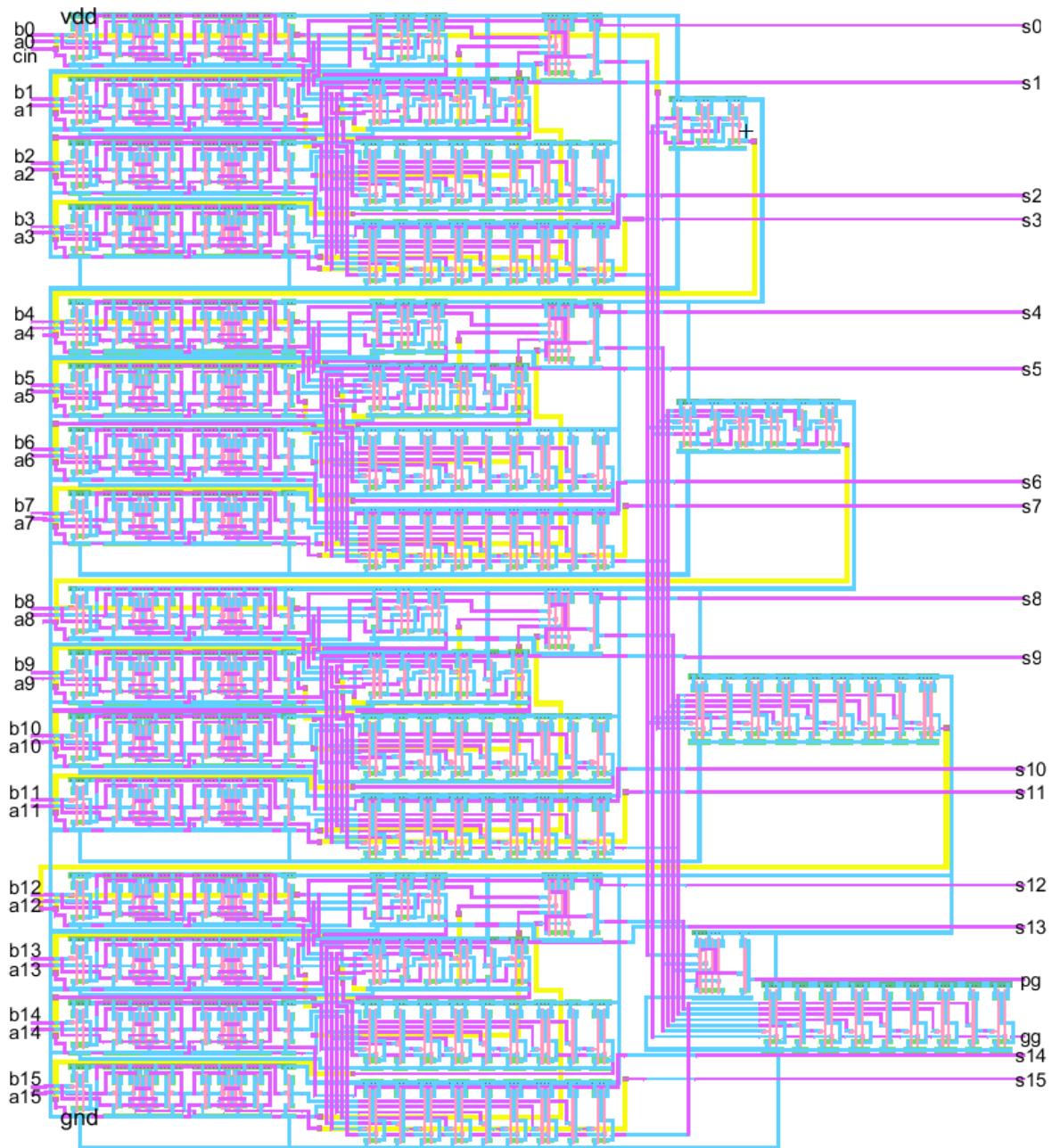
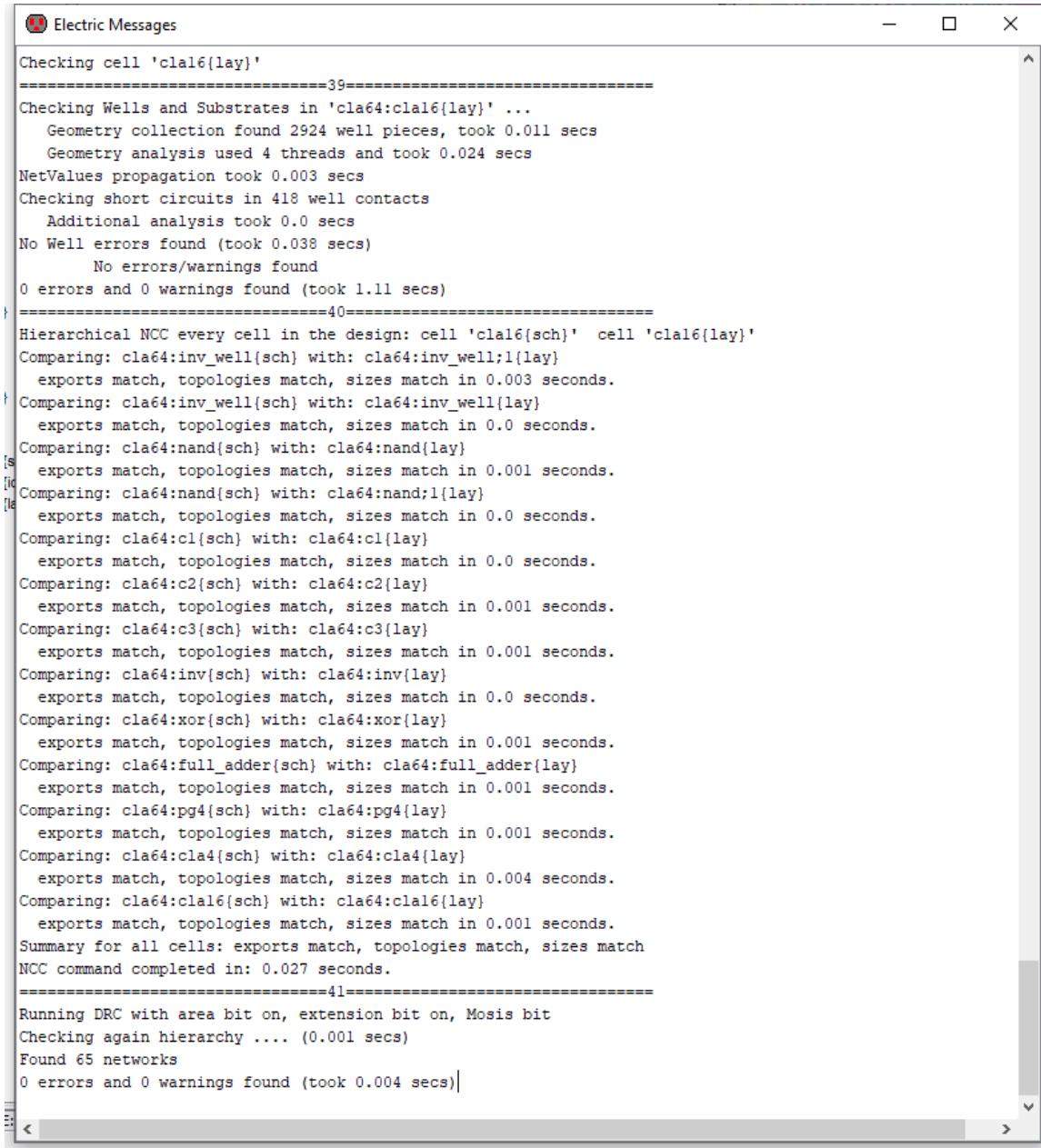


Figure 27: 16-bit CLA Layout



```

Electric Messages

Checking cell 'cla16{lay}'
=====
Checking Wells and Substrates in 'cla64:cla16{lay}' ...
    Geometry collection found 2924 well pieces, took 0.011 secs
    Geometry analysis used 4 threads and took 0.024 secs
NetValues propagation took 0.003 secs
Checking short circuits in 418 well contacts
    Additional analysis took 0.0 secs
No Well errors found (took 0.038 secs)
    No errors/warnings found
0 errors and 0 warnings found (took 1.11 secs)
=====
Hierarchical NCC every cell in the design: cell 'cla16{sch}' cell 'cla16{lay}'
Comparing: cla64:inv_well{sch} with: cla64:inv_well;1{lay}
    exports match, topologies match, sizes match in 0.003 seconds.
Comparing: cla64:inv_well{sch} with: cla64:inv_well{lay}
    exports match, topologies match, sizes match in 0.0 seconds.
Comparing: cla64:nand{sch} with: cla64:nand{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:nand{sch} with: cla64:nand;1{lay}
    exports match, topologies match, sizes match in 0.0 seconds.
Comparing: cla64:c1{sch} with: cla64:c1{lay}
    exports match, topologies match, sizes match in 0.0 seconds.
Comparing: cla64:c2{sch} with: cla64:c2{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:c3{sch} with: cla64:c3{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:inv{sch} with: cla64:inv{lay}
    exports match, topologies match, sizes match in 0.0 seconds.
Comparing: cla64:xor{sch} with: cla64:xor{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:full_adder{sch} with: cla64:full_adder{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:pg4{sch} with: cla64:pg4{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:cla4{sch} with: cla64:cla4{lay}
    exports match, topologies match, sizes match in 0.004 seconds.
Comparing: cla64:cla16{sch} with: cla64:cla16{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Summary for all cells: exports match, topologies match, sizes match
NCC command completed in: 0.027 seconds.
=====
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.001 secs)
Found 65 networks
0 errors and 0 warnings found (took 0.004 secs)

```

**Figure 28: 16-bit CLA Checks**

On the next page is the final layout of the 32-bit CLA. Due to the sheer size of the layout, it's split into two figures—one for the top part and one for the bottom.

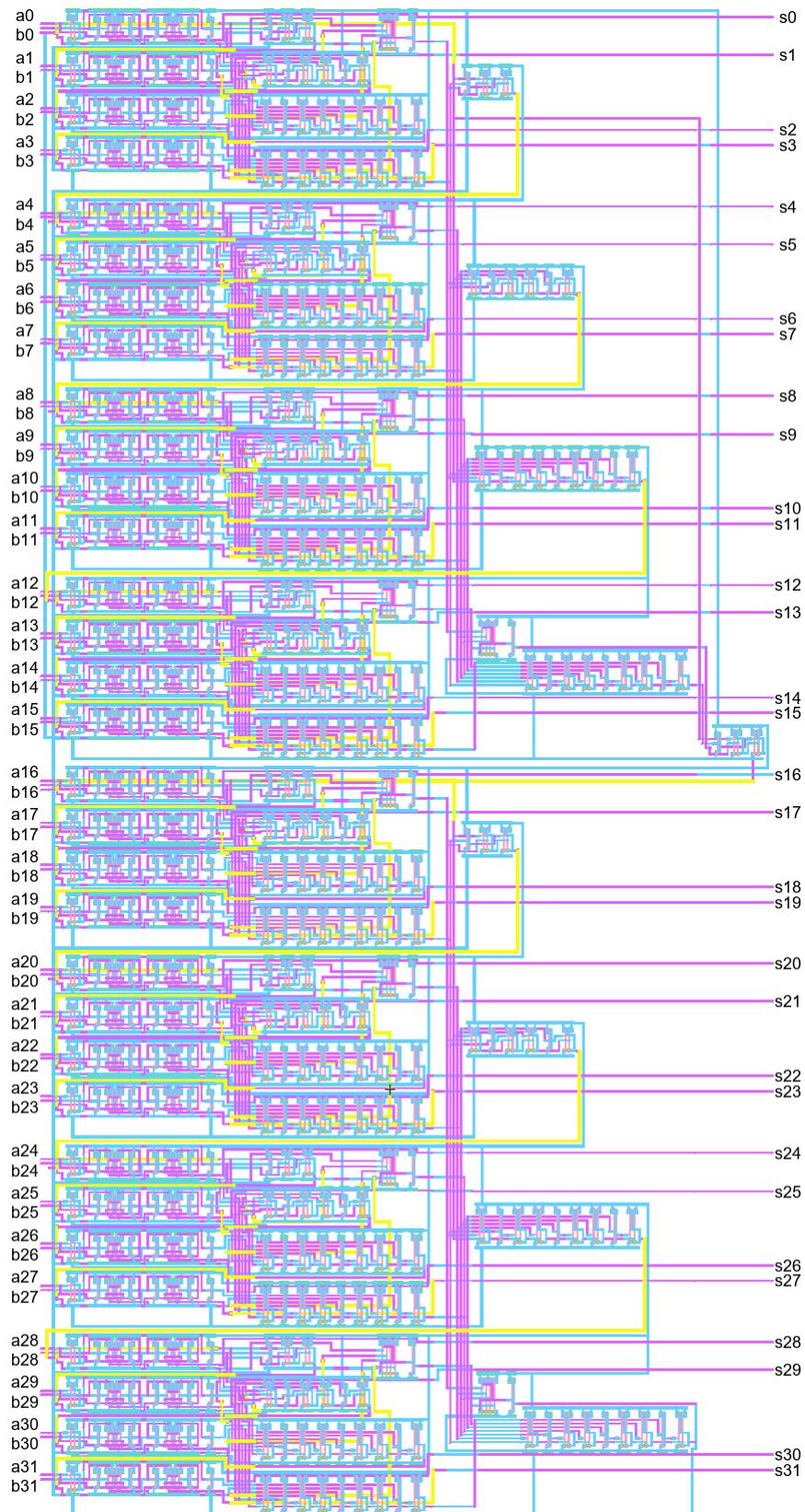


Figure 29: 32-bit CLA Layout (Full)

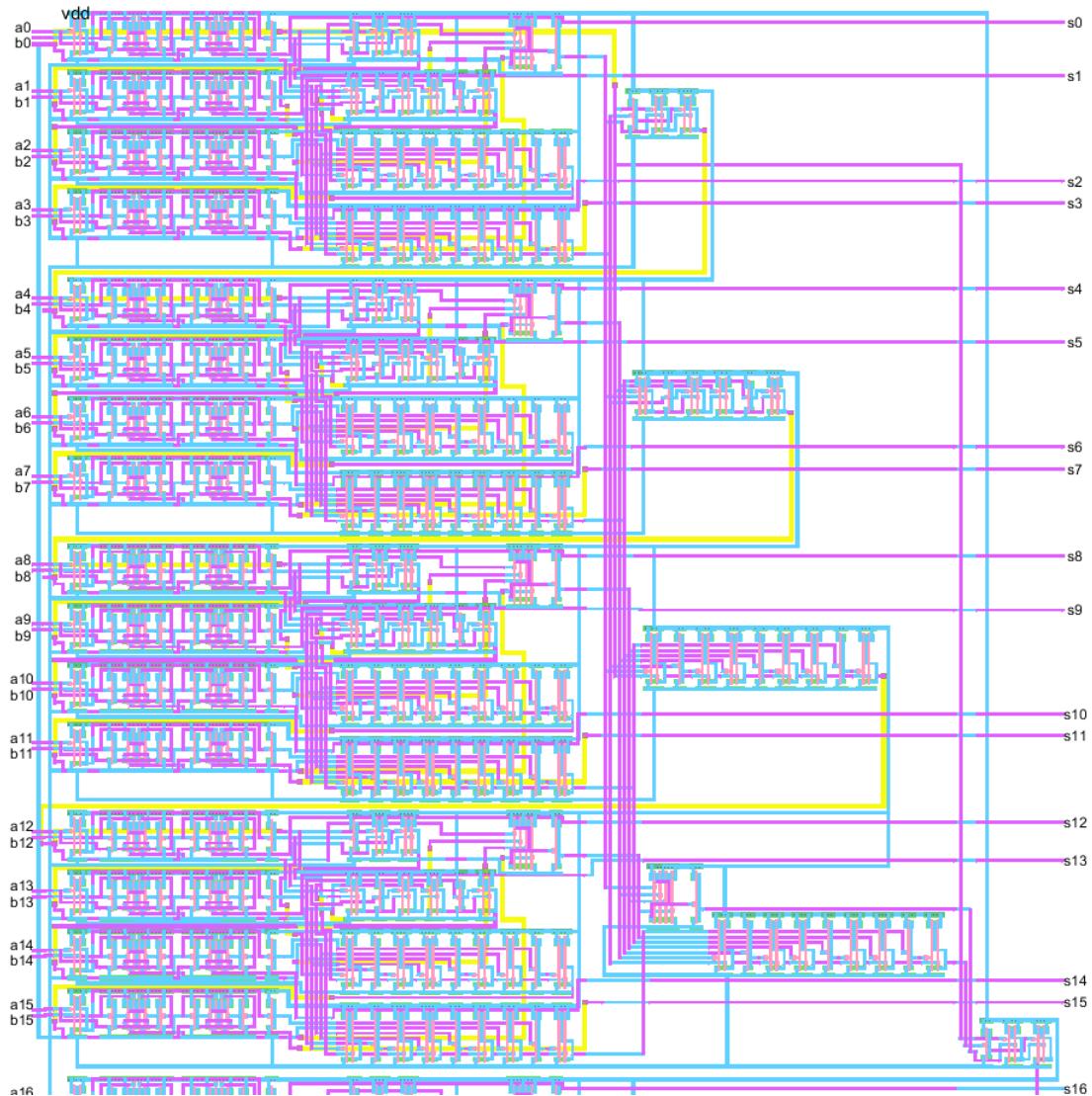


Figure 30: 32-bit CLA Layout (Top)

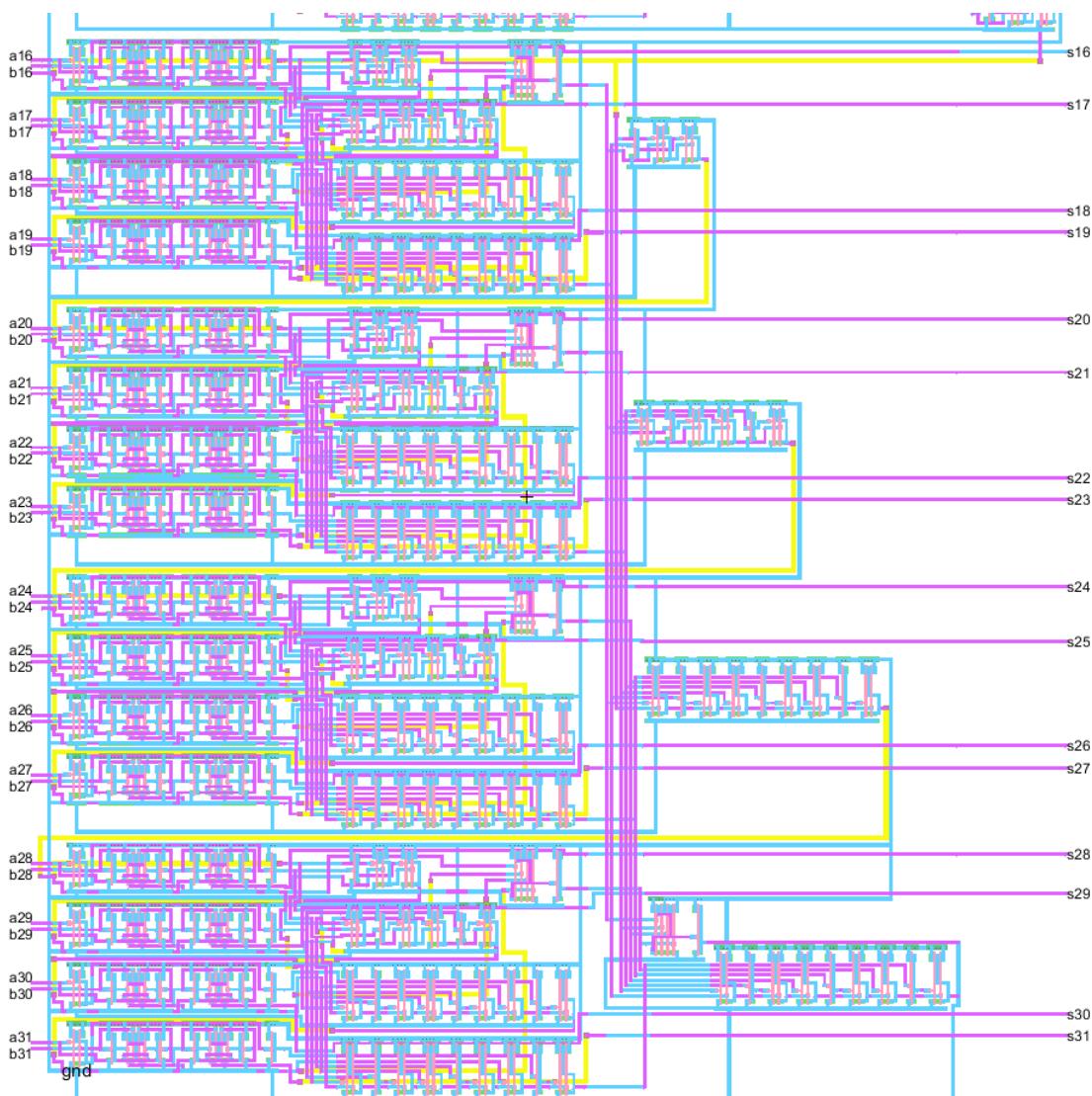


Figure 31: 32-bit CLA Layout (Bottom)

**Electric Messages**

```
=====
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.001 secs)
Found 102 networks
Checking cell 'cla32{lay}'
    No errors/warnings found
0 errors and 0 warnings found (took 3.144 secs)
=====
43=====
Checking Wells and Substrates in 'cla64:cla32{lay}' ...
    Geometry collection found 5880 well pieces, took 0.008 secs
    Geometry analysis used 4 threads and took 0.003 secs
NetValues propagation took 0.004 secs
Checking short circuits in 842 well contacts
    Additional analysis took 0.001 secs
No Well errors found (took 0.016 secs)
=====
44=====
Hierarchical NCC every cell in the design: cell 'cla32{sch}'  cell 'cla32{lay}'
Comparing: cla64:inv_well{sch} with: cla64:inv_well;l{lay}
    exports match, topologies match, sizes match in 0.0 seconds.
Comparing: cla64:inv_well{sch} with: cla64:inv_well{lay}
    exports match, topologies match, sizes match in 0.002 seconds.
Comparing: cla64:nand{sch} with: cla64:nand;l{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:nand{sch} with: cla64:nand{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:c1{sch} with: cla64:c1{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:c2{sch} with: cla64:c2{lay}
    exports match, topologies match, sizes match in 0.0 seconds.
Comparing: cla64:c3{sch} with: cla64:c3{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:inv{sch} with: cla64:inv{lay}
    exports match, topologies match, sizes match in 0.0 seconds.
Comparing: cla64:xor{sch} with: cla64:xor{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:full_adder{sch} with: cla64:full_adder{lay}
    exports match, topologies match, sizes match in 0.0 seconds.
Comparing: cla64:pg4{sch} with: cla64:pg4{lay}
    exports match, topologies match, sizes match in 0.003 seconds.
Comparing: cla64:cla4{sch} with: cla64:cla4{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:cla16{sch} with: cla64:cla16{lay}
    exports match, topologies match, sizes match in 0.001 seconds.
Comparing: cla64:cla32{sch} with: cla64:cla32{lay}
    exports match, topologies match, sizes match in 0.002 seconds.
Summary for all cells: exports match, topologies match, sizes match
NCC command completed in: 0.019 seconds.
```

Figure 32: 32-bit CLA Checks

## Section 5: IRSIM Logic Simulations and Measurements

IRSIM is used to test the logic behind the circuit that we made and to verify that it satisfies our design requirements. In this section, we would also be discussing the results of the IRSIM simulations. The inputs that were given to IRSIM are shown in the table below. We would first test five additions,  $A + B = S$ , and later test five subtractions, using the results from the addition. The subtraction would then be  $S - B = A$ , so as a result, we should get our initial input back. Using this sort of computation makes verifying our circuit easier, in addition to testing the logic of the circuit.

**Table 1: Input and Outputs of the Computations (IRSIM)**

	<b>Input: A (Hexadecimal/Decimal)</b>	<b>Input: B (Hexadecimal/Decimal)</b>	<b>Expected Output: S (Hexadecimal/Decimal)</b>
<b>First Addition</b>	0x7FFFFFFF	0x00000001	0x80000000
	2147483647	1	2147483648
<b>Second Addition</b>	0x7FFFFFFF	0x80000000	0xFFFFFFFF
	2147483647	2147483648	4294967295
<b>Third Addition</b>	0x12162019	0x00040000	0x121A2019
	303439897	262144	303702041
<b>Fourth Addition</b>	0x11031997	0x01130682	0x12162019
	285415831	18024066	303439897
<b>Fifth Addition</b>	0x09161998	0x09000681	0x12162019
	152443288	150996609	303439897
<b>First Subtraction</b>	0x80000000	-0x00000001 =	0x7FFFFFFF
	2147483648	0xFFFFFFFF	2147483647
<b>Second Subtraction</b>	0xFFFFFFFF	-0x80000000 =	0x7FFFFFFF
	4294967295	0x80000000	2147483647
		-2147483648	
<b>Third Subtraction</b>	0x121A2019	-0x00040000 =	0x12162019
	303702041	0xFFFFC0000	303439897
		-262144	
<b>Fourth Subtraction</b>	0x12162019	-0x01130682 =	0x11031997
	303439897	0xFEECF97E	285415831
		-18024066	
<b>Fifth Subtraction</b>	0x12162019	-0x09000681 =	0x09161998
	303439897	0xF6FFF97F	152443288
		-150996609	

### Section 5.1: Schematic

The input and output waveforms generated by IRSIM for the 32-bit CLA schematic are shown below. Each figure shows 8 bits each. Figures 33 to 44 show the inputs and outputs for the five additions. Figures 45 to 56 show the inputs and outputs for the five subtractions.

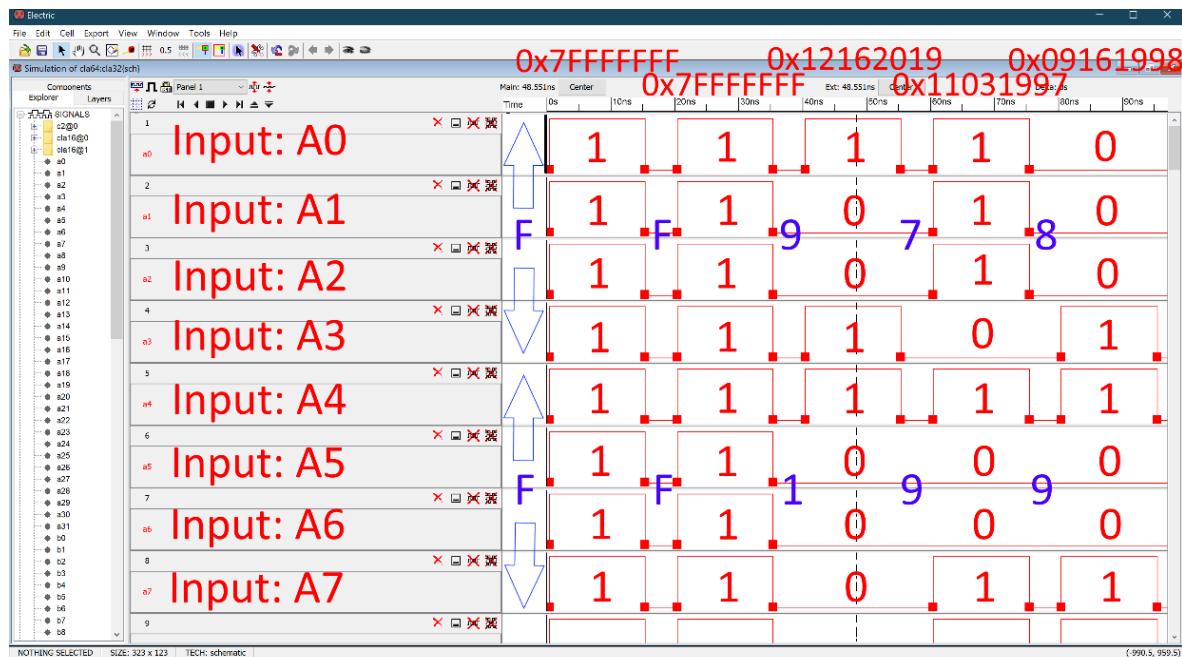


Figure 33: Input A[0:7]

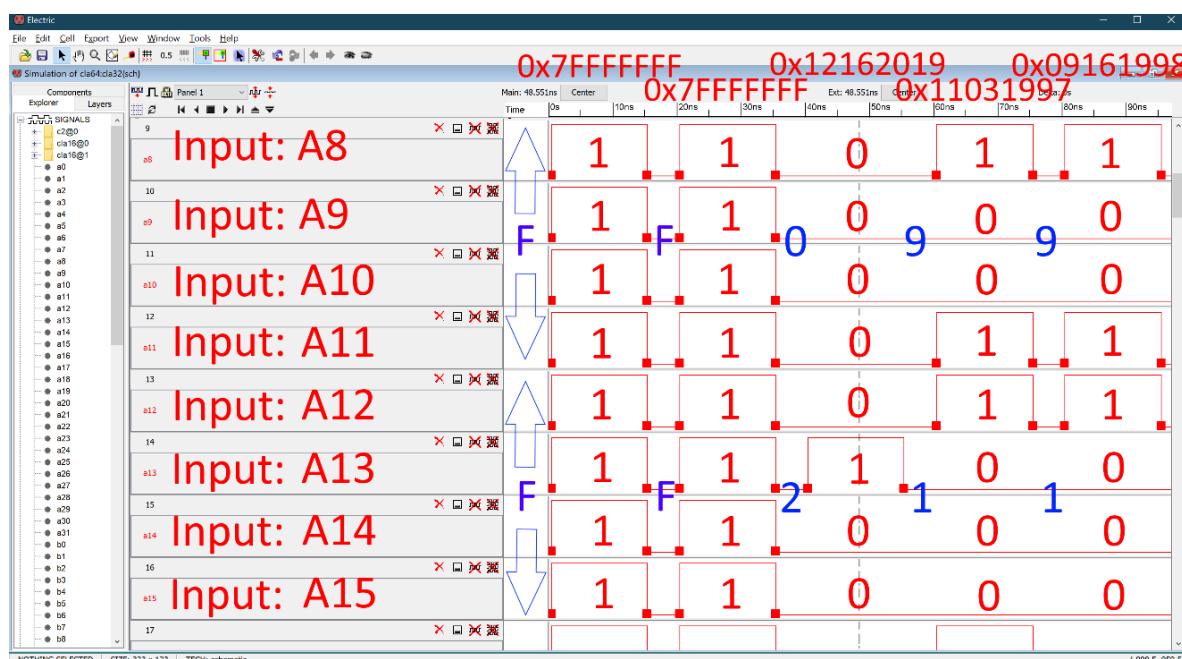


Figure 34: Input A[8:15]

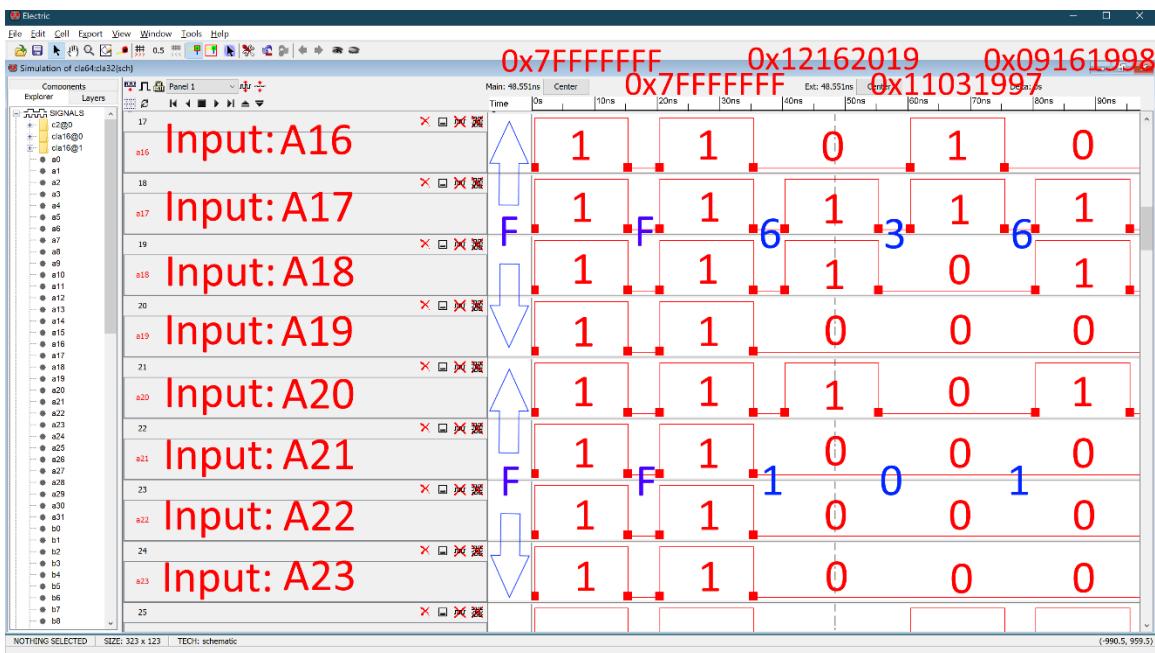


Figure 35: Input A[16:23]

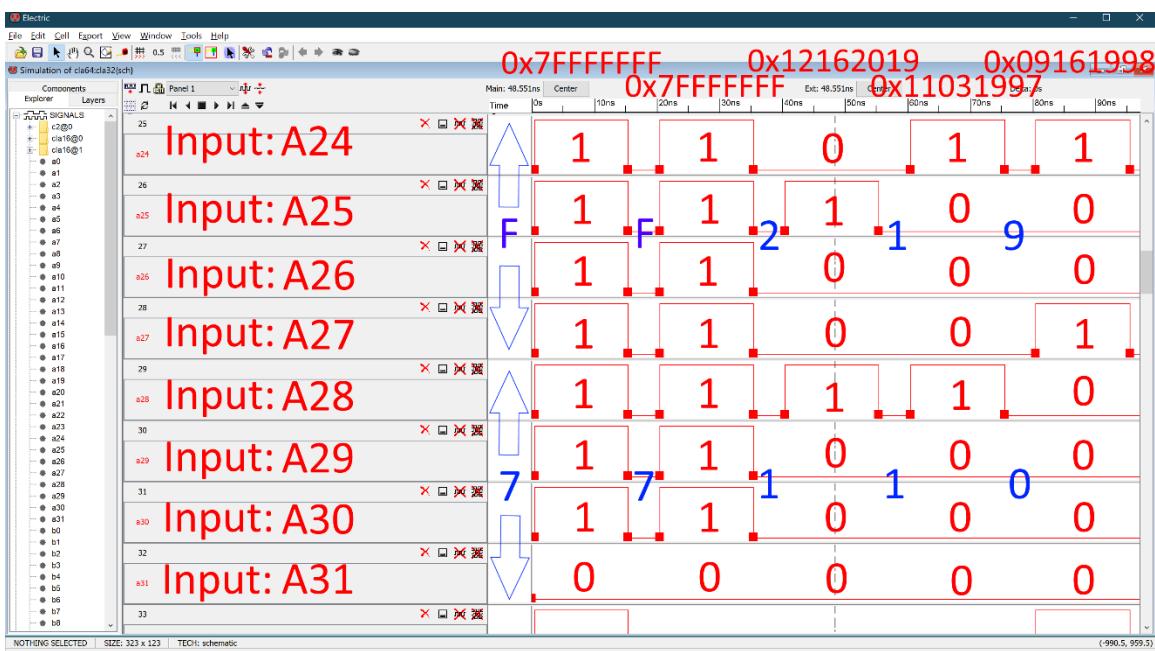


Figure 36: Input A[24:31]

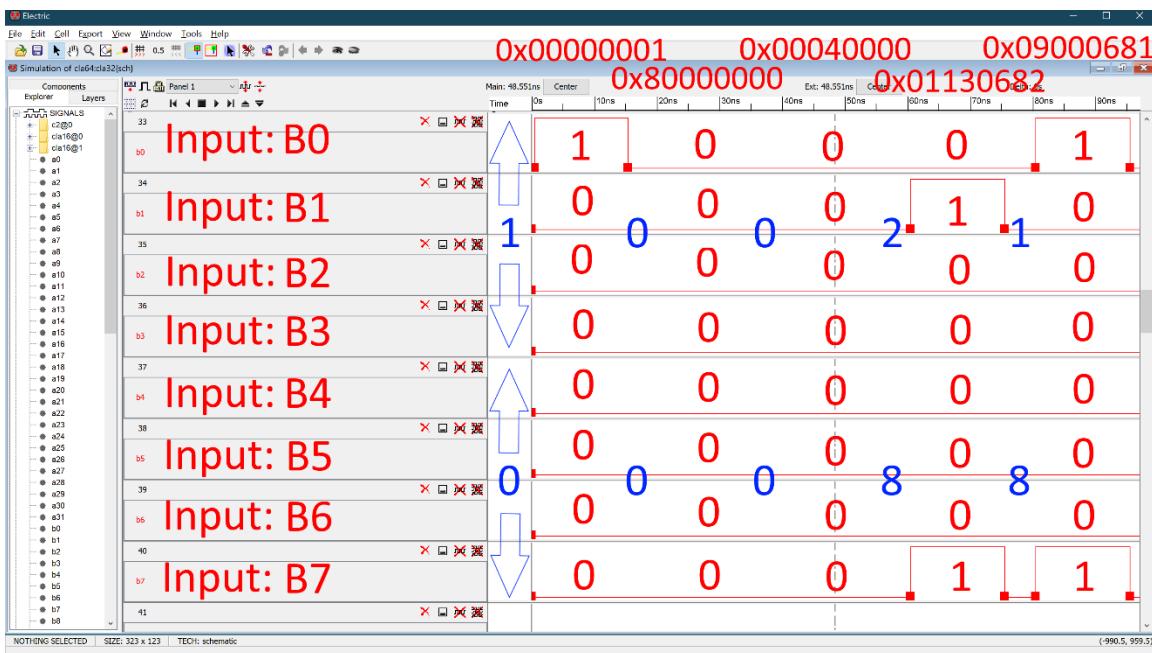


Figure 37: Input B[0:7]

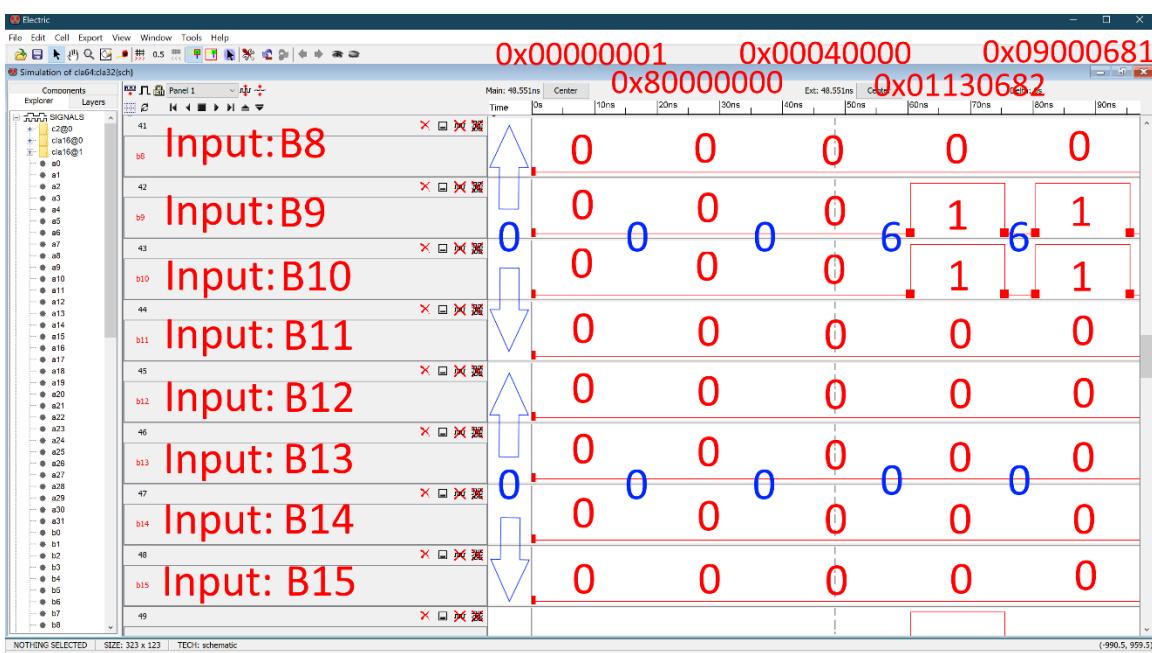


Figure 38: Input B[8:15]

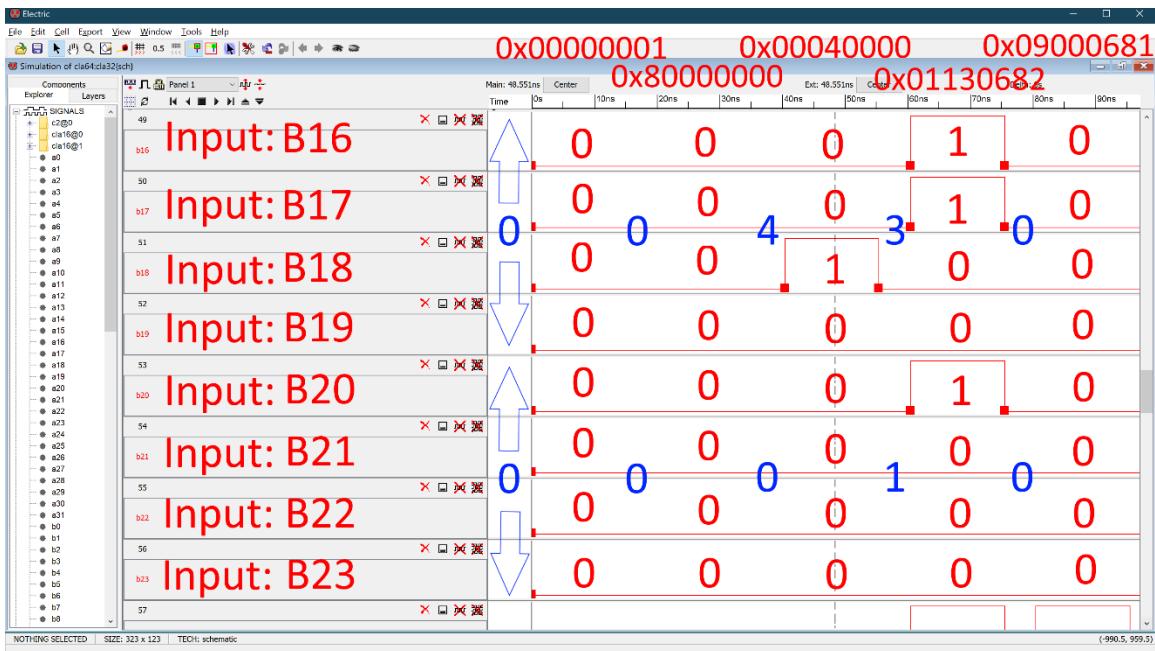


Figure 39: Input B[16:23]

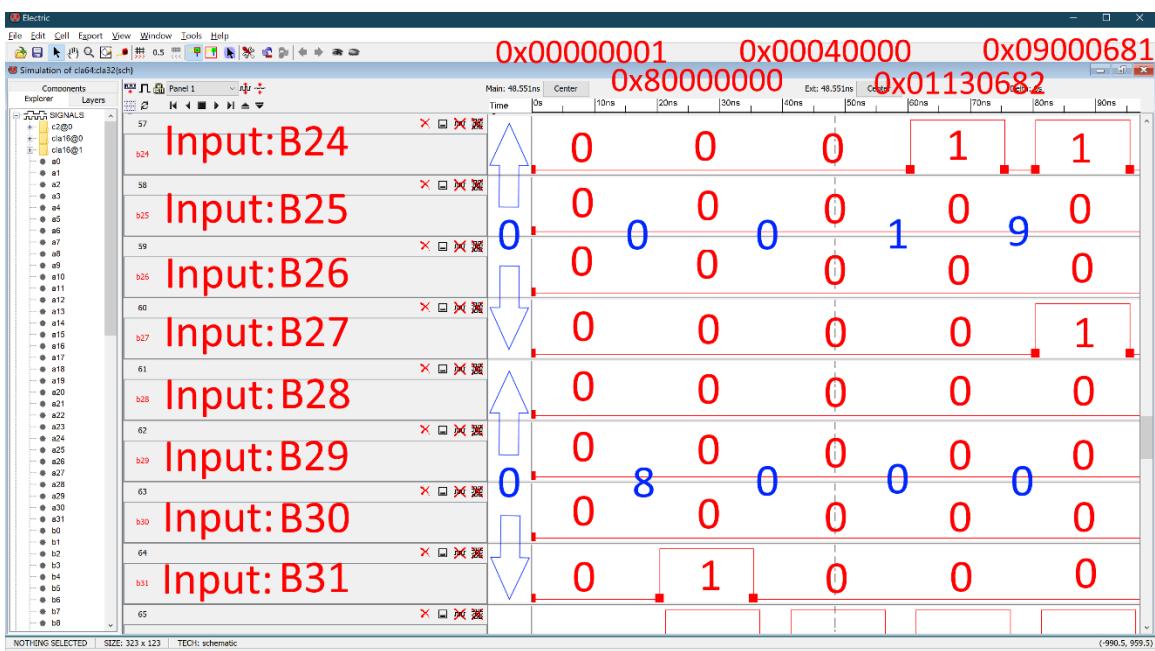


Figure 40: Input B[24:31]

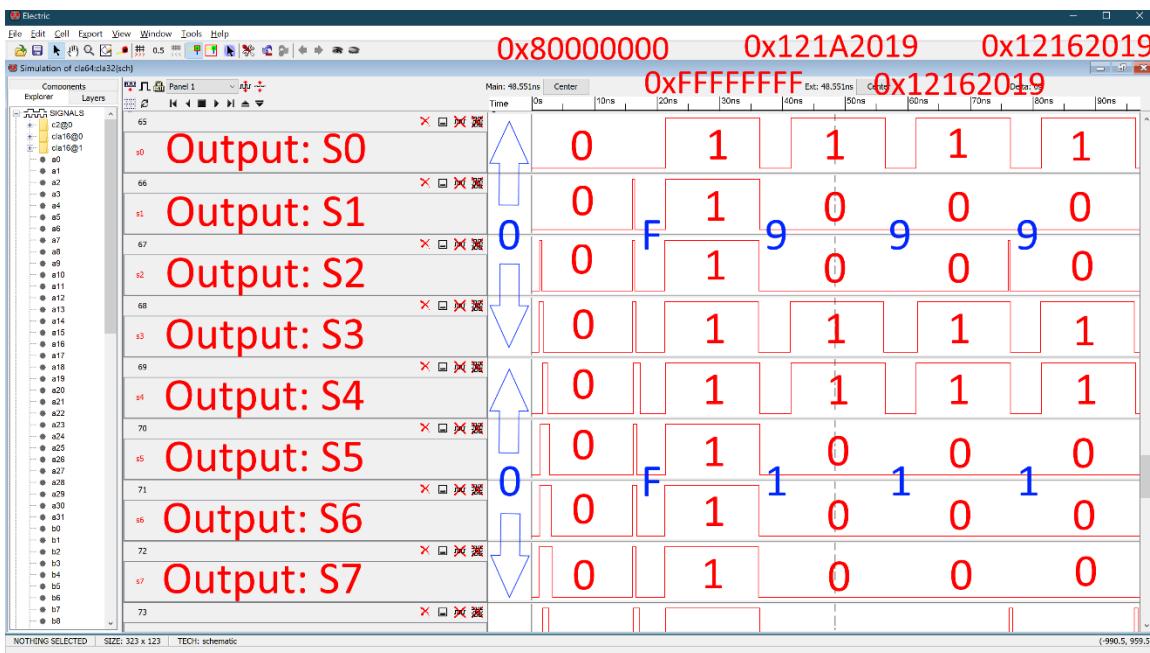


Figure 41: Addition Schematic IRSIM - Output S[0:7]

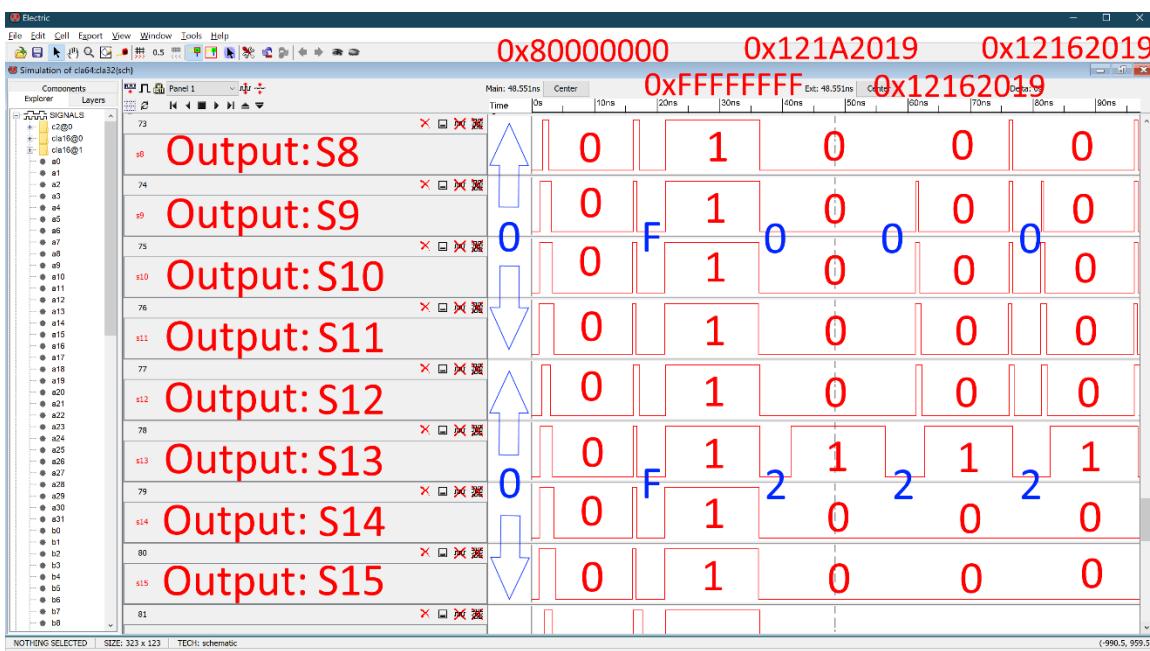


Figure 42: Addition Schematic IRSIM - Output S[8:15]

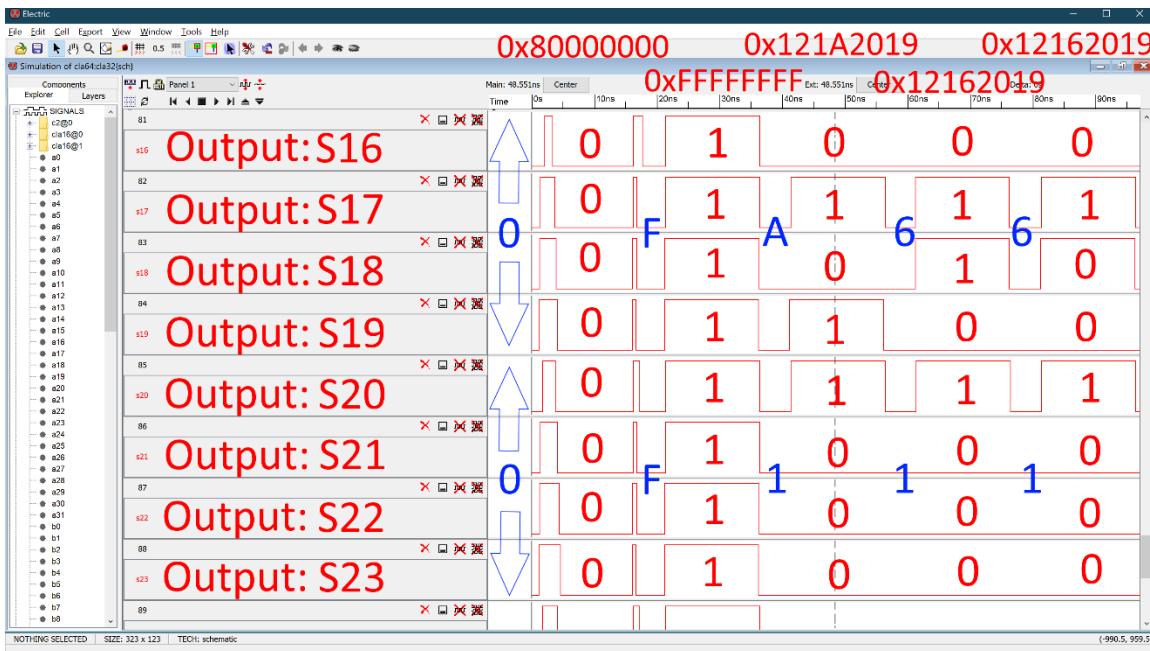


Figure 43: Addition Schematic IRSIM - Output S[16:23]

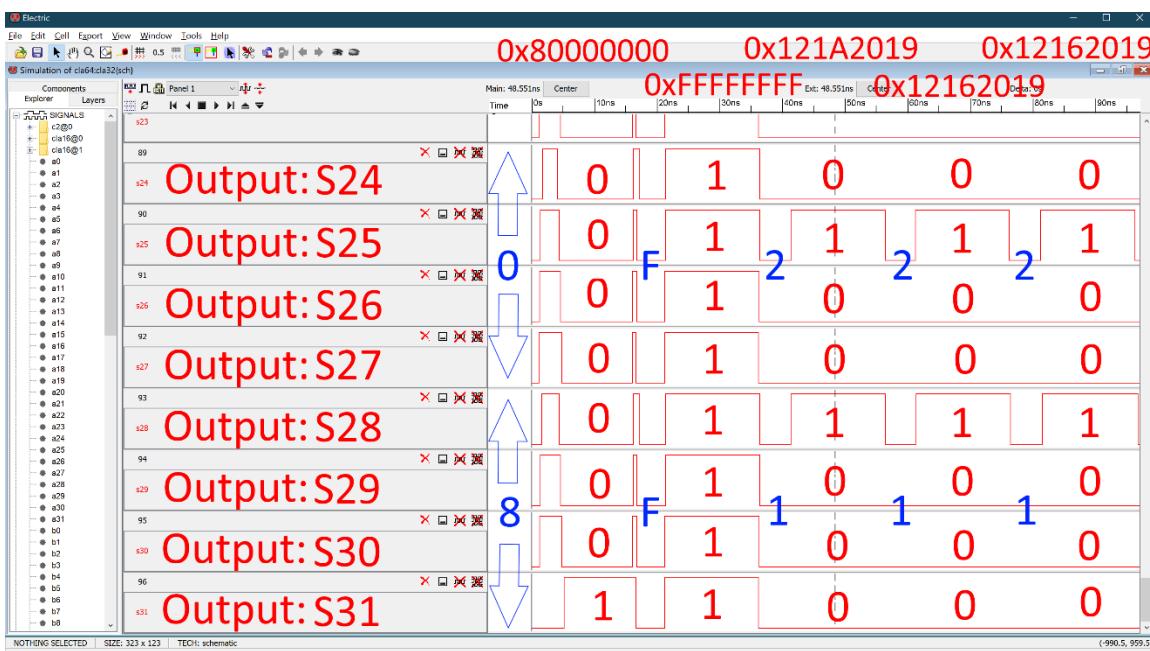


Figure 44: Addition Schematic IRSIM - Output S[24:31]

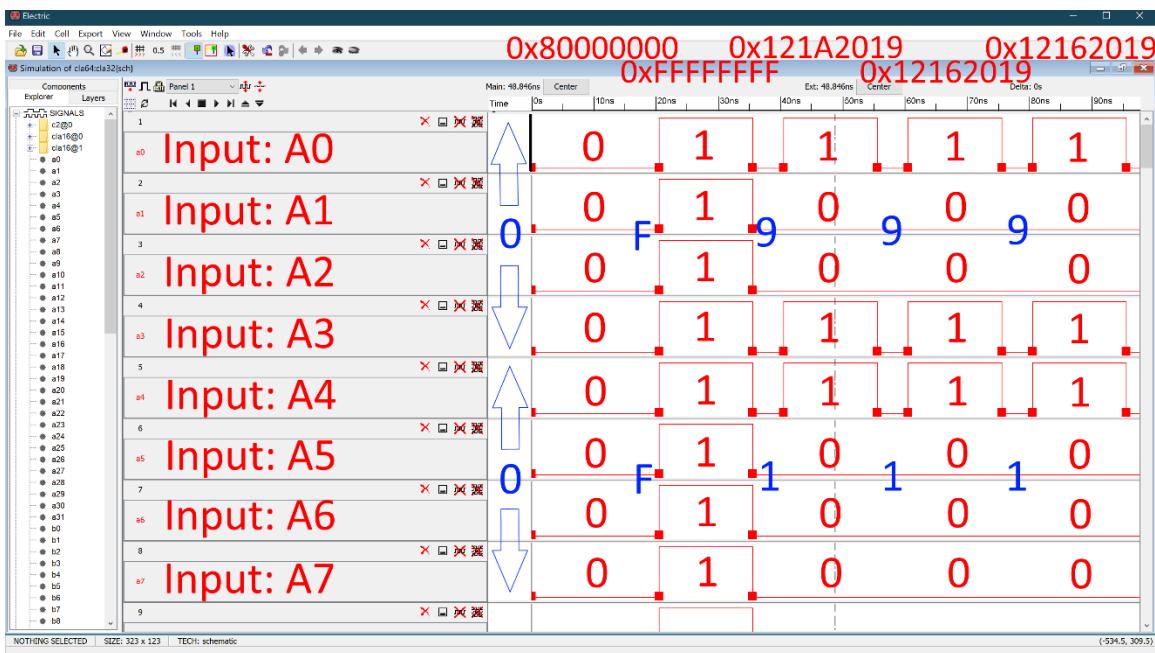


Figure 45: Input A[0:7]

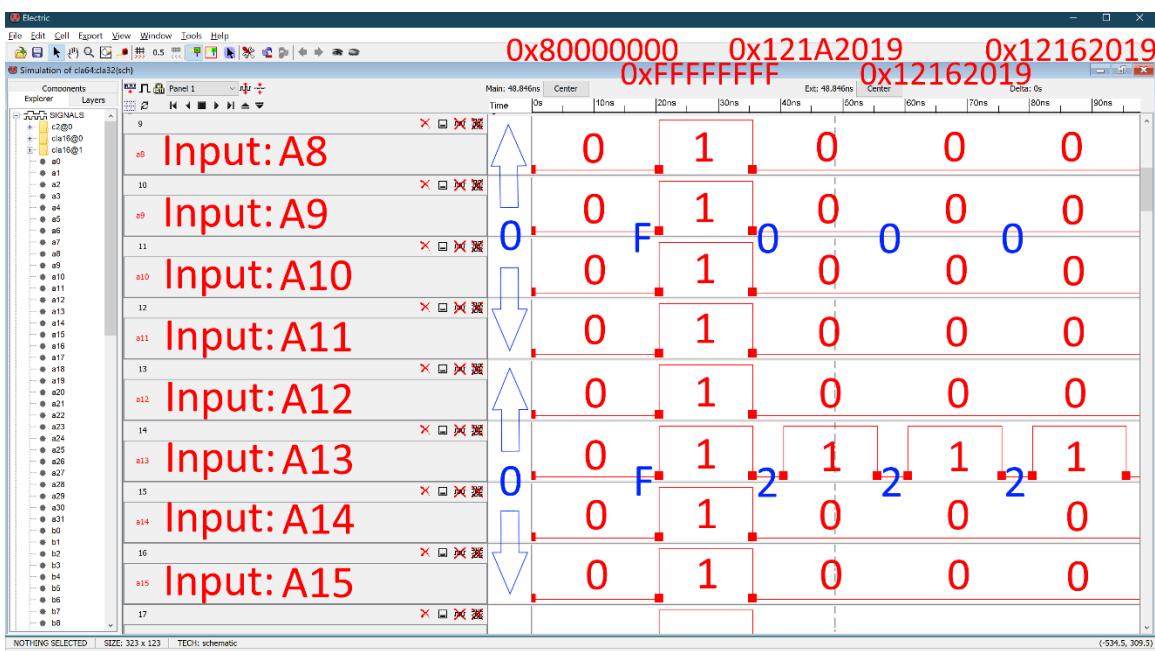


Figure 46: Input A[15:8]

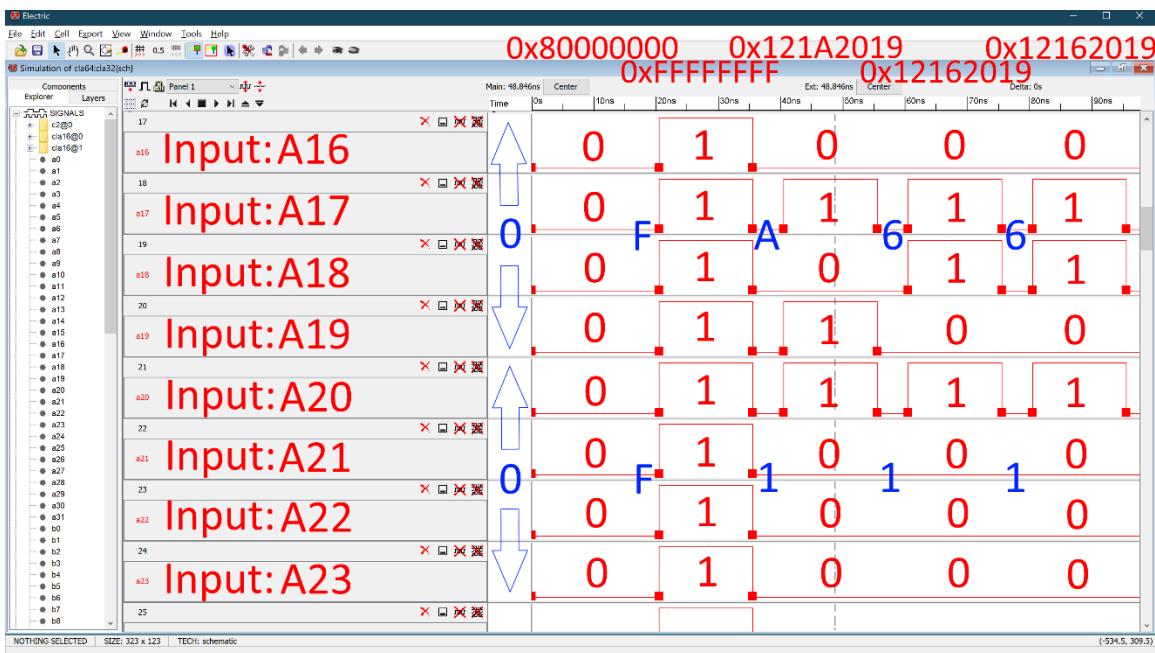


Figure 47: Input A[16:23]

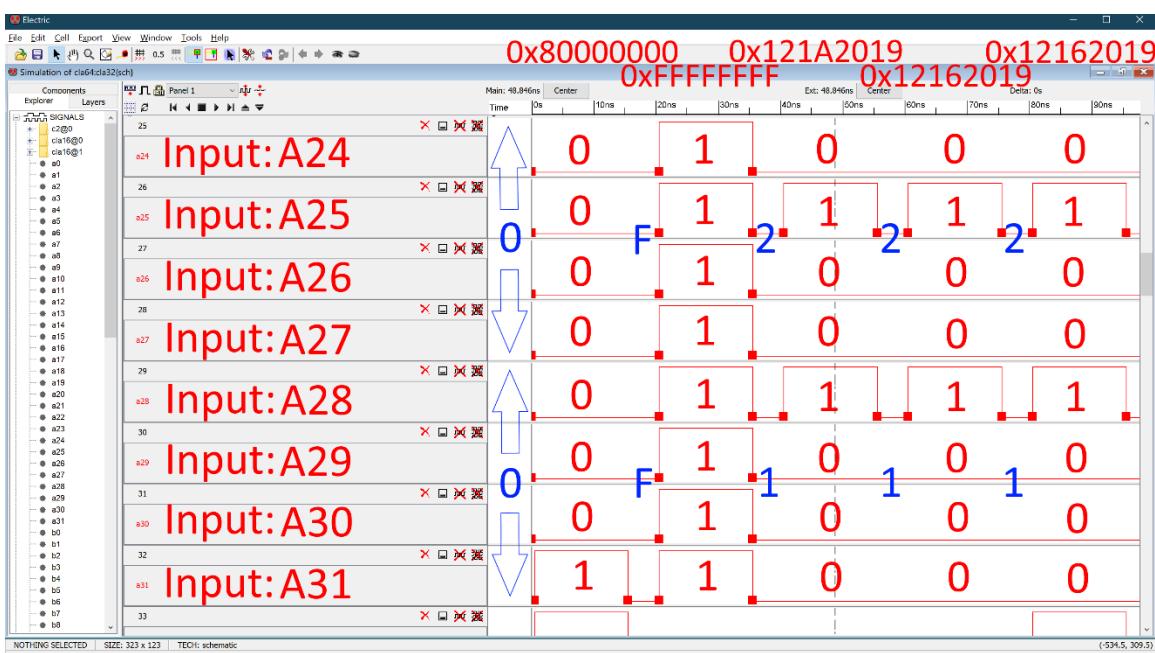


Figure 48: Input A[24:31]

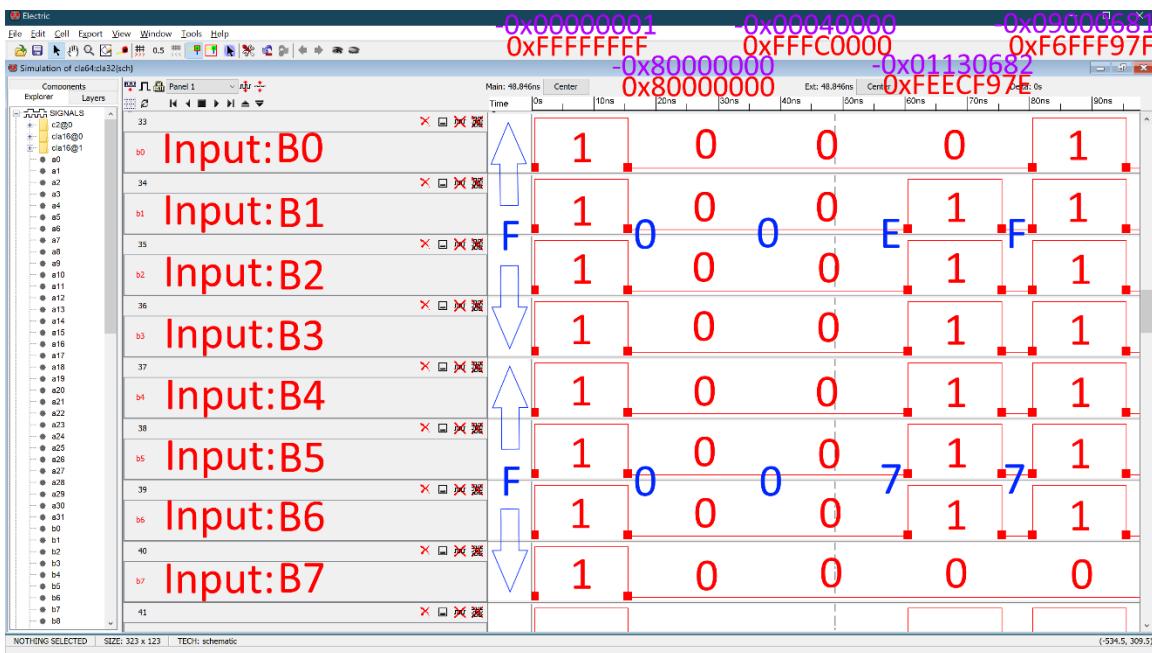


Figure 49: Input B[0:7]

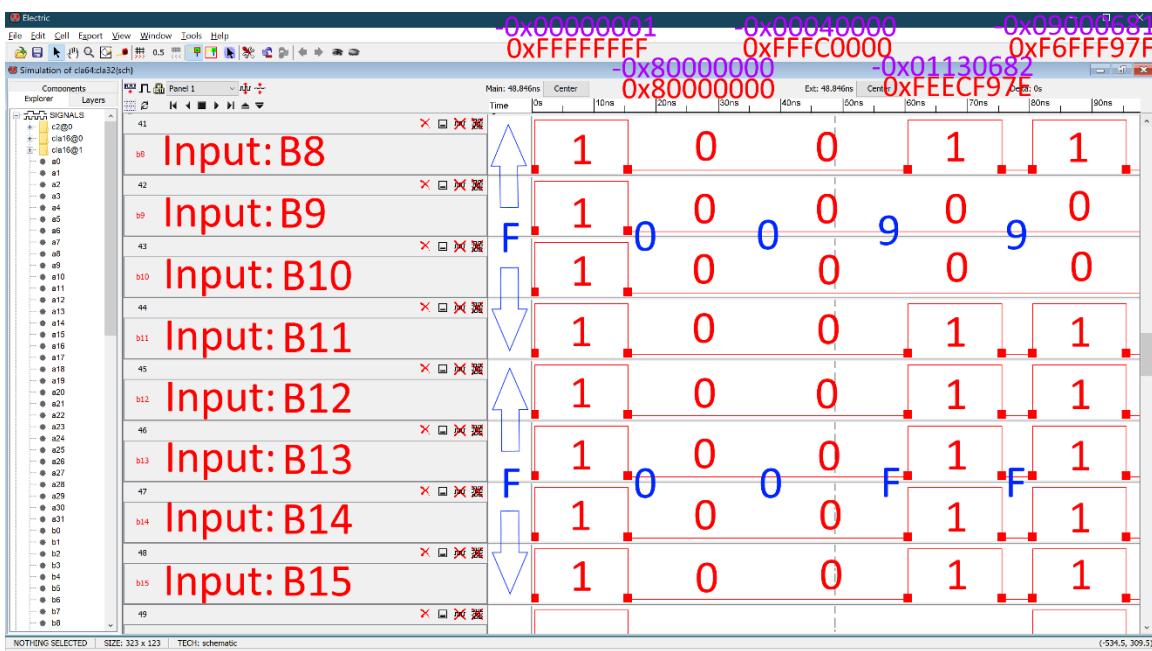


Figure 50: Input B[8:15]

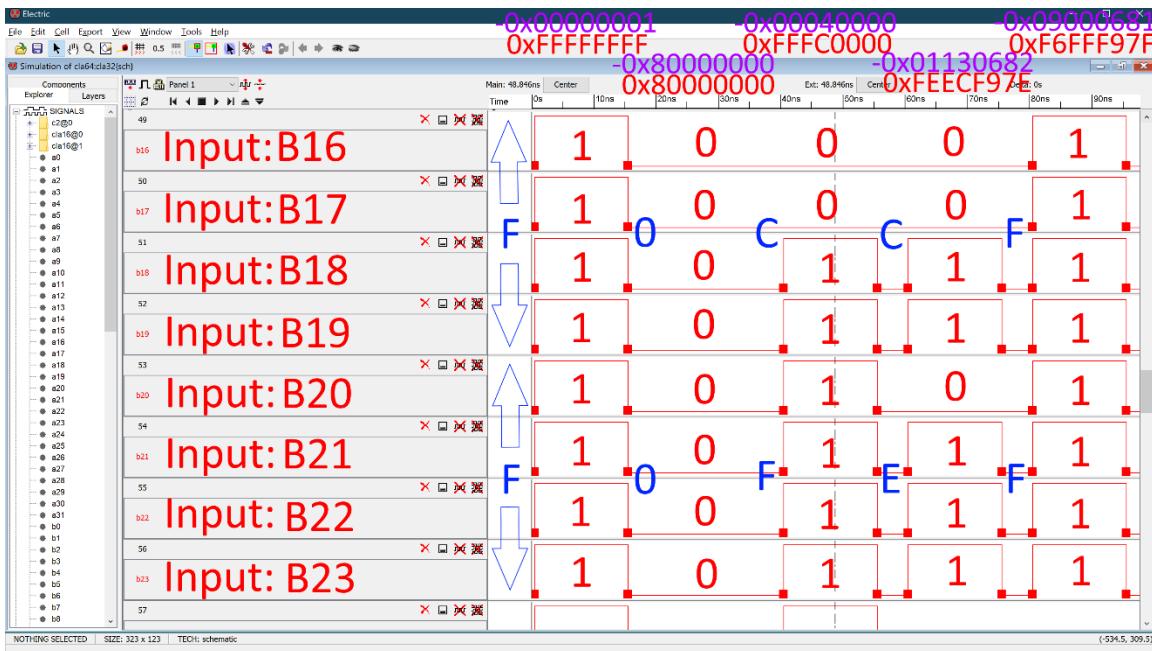


Figure 51: Input B[16:23]

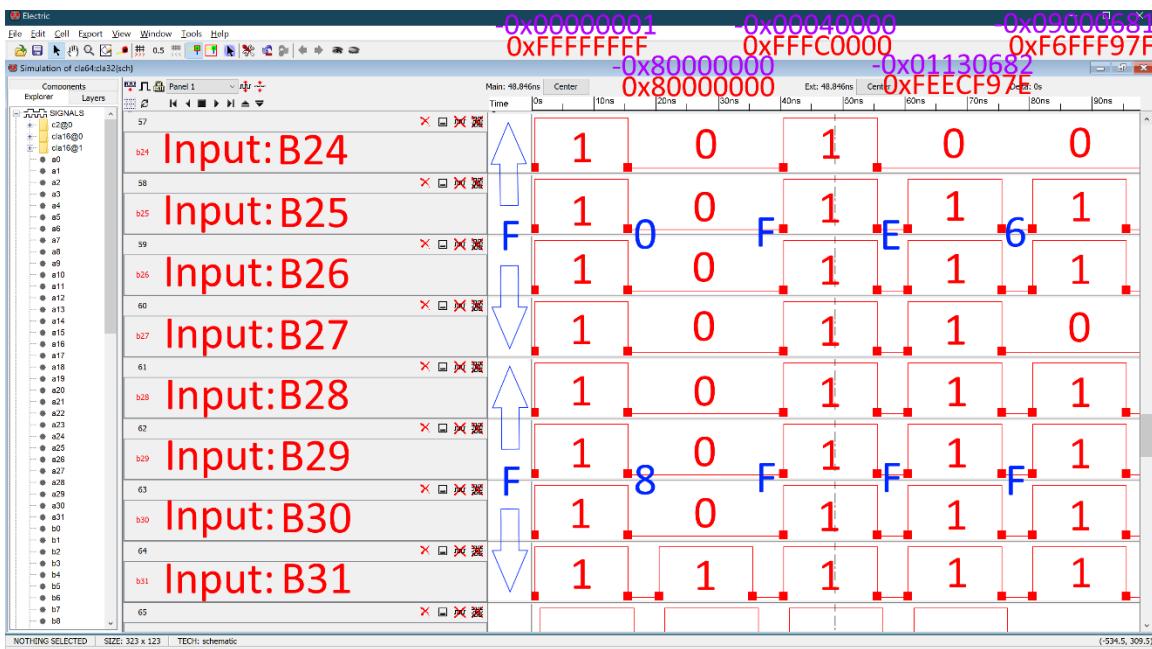


Figure 52: Input B[24:31]

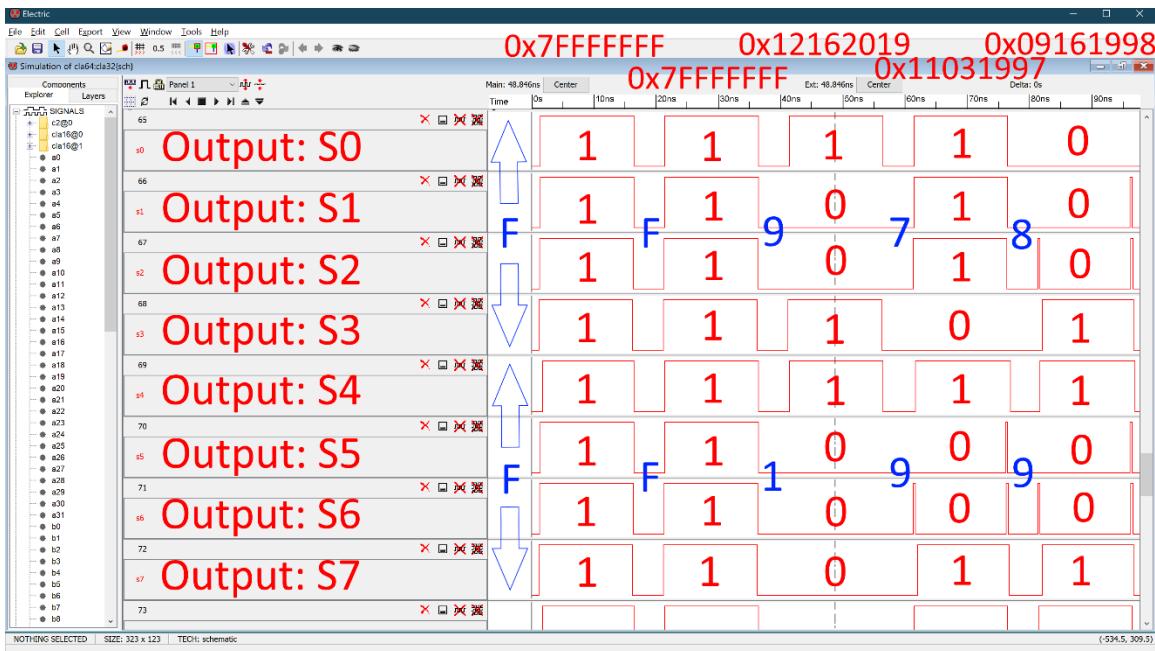


Figure 53: Subtraction Schematic IRSIM - Output S[0:7]

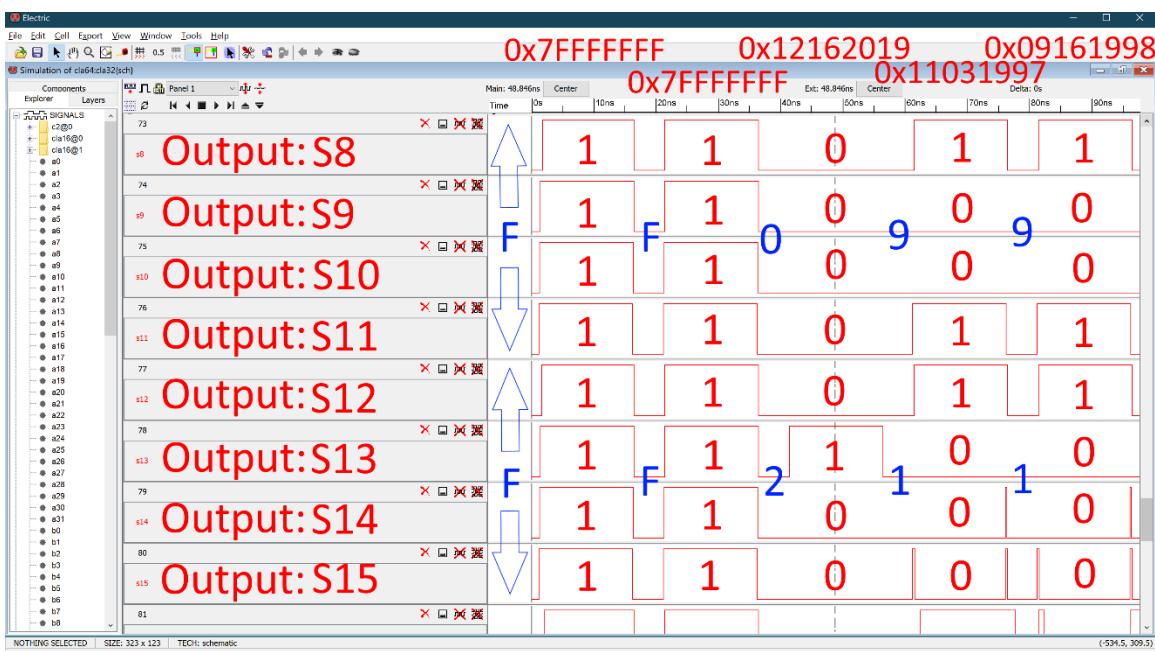


Figure 54: Subtraction Schematic IRSIM - Output S[8:15]

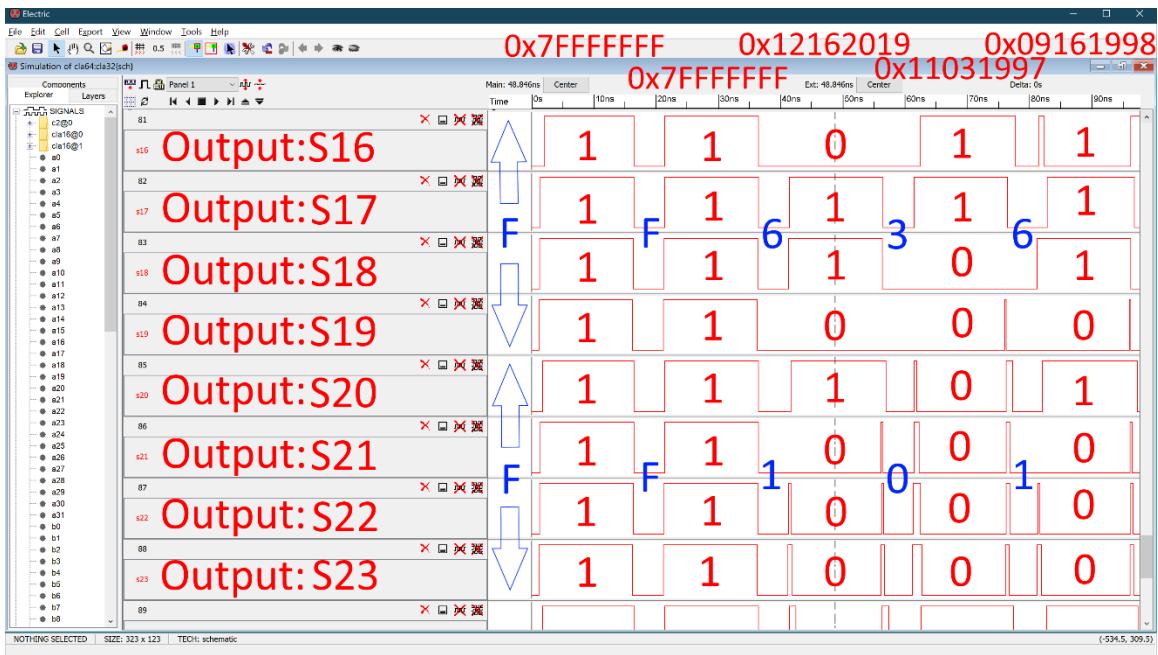


Figure 55: Subtraction Schematic IRSIM - Output S[16:23]

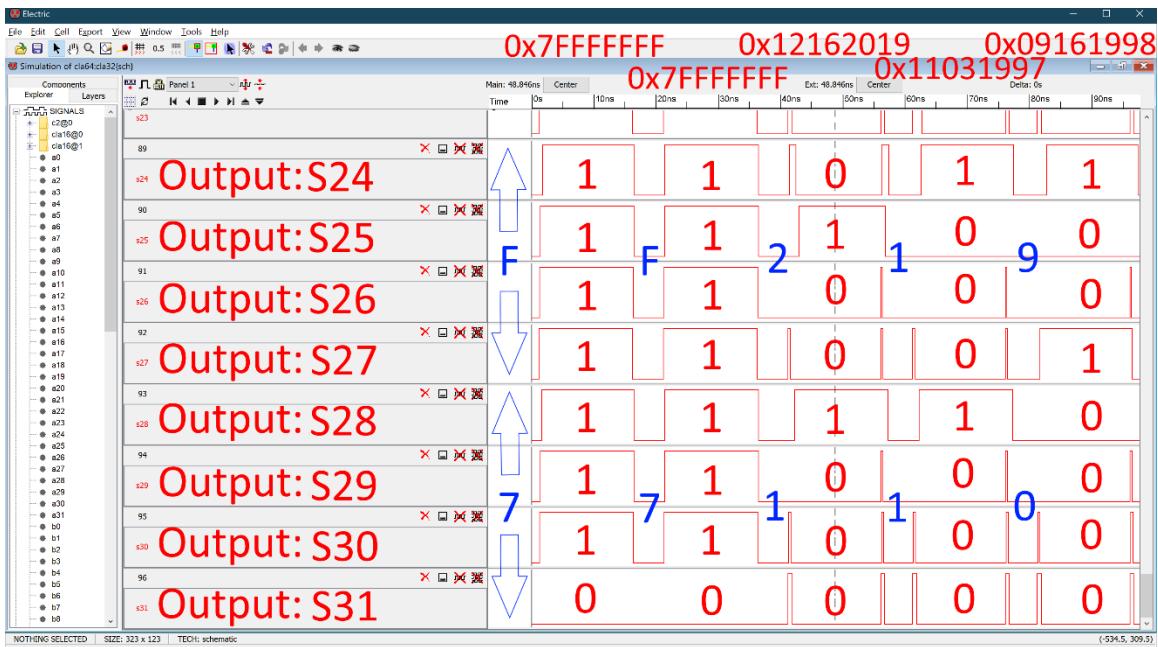
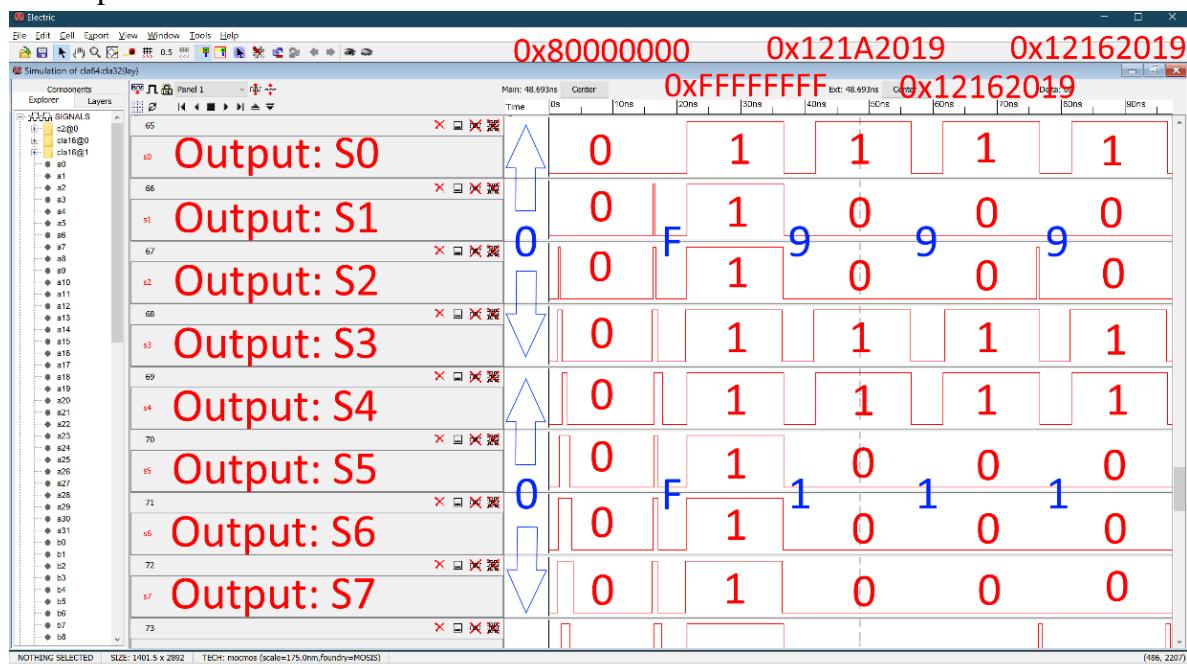


Figure 56: Subtraction Schematic IRSIM - Output S[24:31]

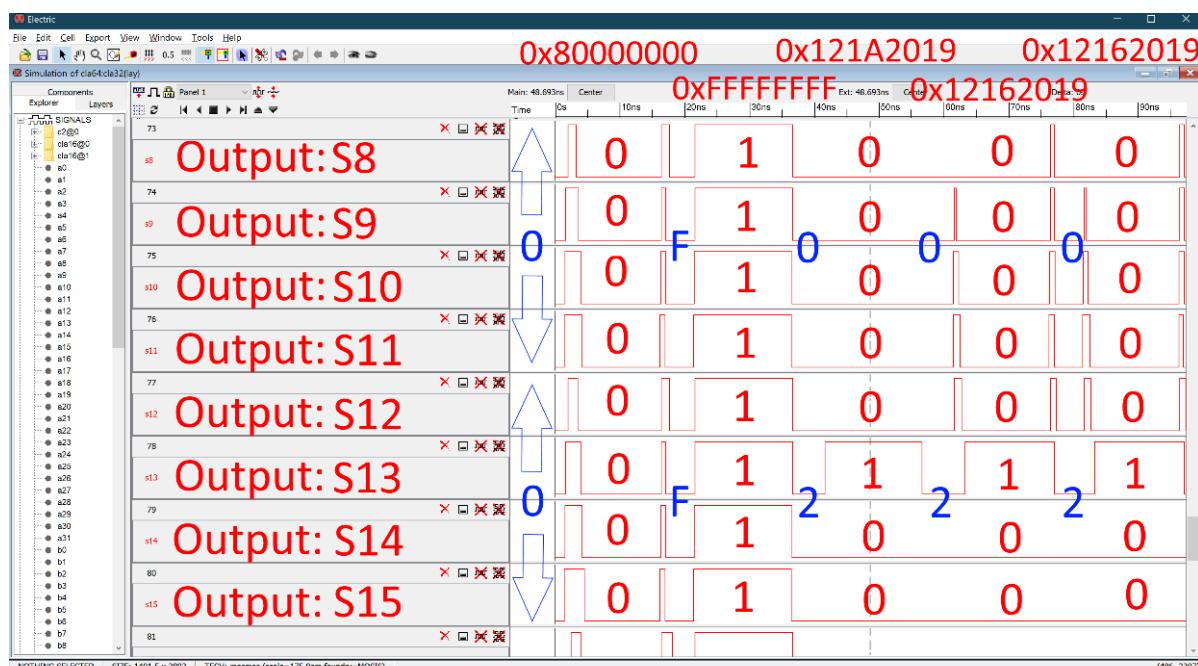
The output matches what we expected so we can confirm that the schematic satisfies our design requirements. In addition, it should be noted that there are some spikes before we get the correct output, and that is due to the delay of the design.

## Section 5.2: Layout

Below are the waveforms generated by IRSIM for the 32-bit CLA layout. Because the same input waveforms are used, only the output sum bits are shown for the layout. Figures 57 to 60 shows the outputs for the five additions. Figures 61 to 64 shows the outputs for the five subtractions.



**Figure 57: Addition Layout IRSIM - Output S[0:7]**



**Figure 58: Addition Layout IRSIM - Output S[8:15]**

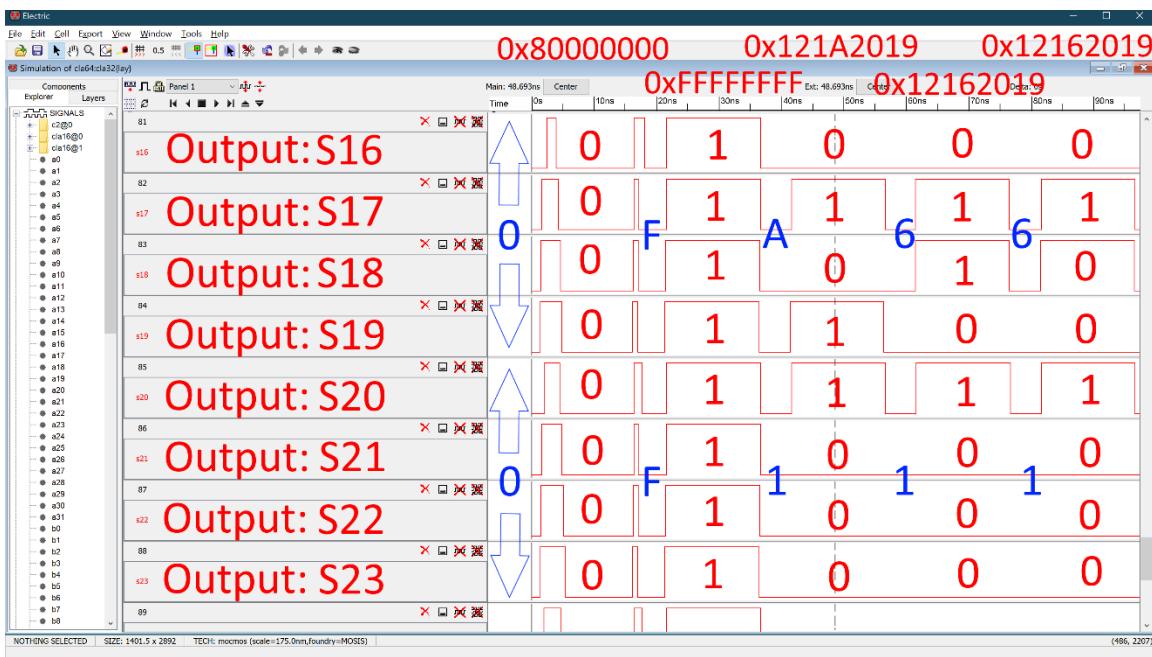


Figure 59: Addition Layout IRSIM - Output S[16:23]

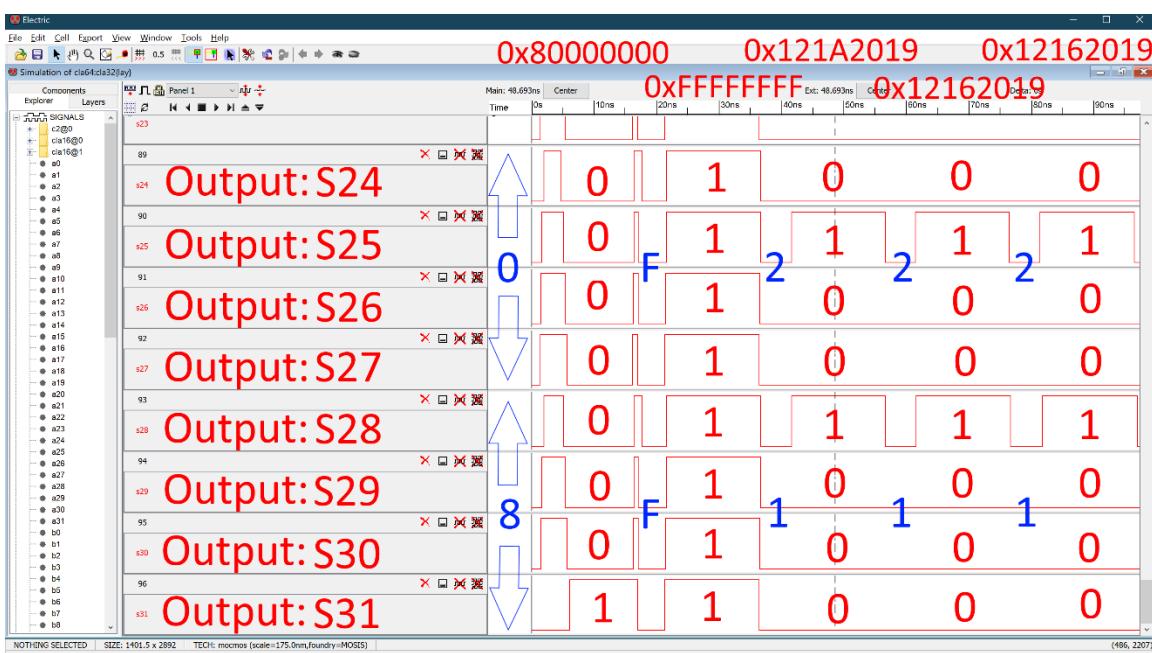


Figure 60: Addition Layout IRSIM - Output S[24:31]

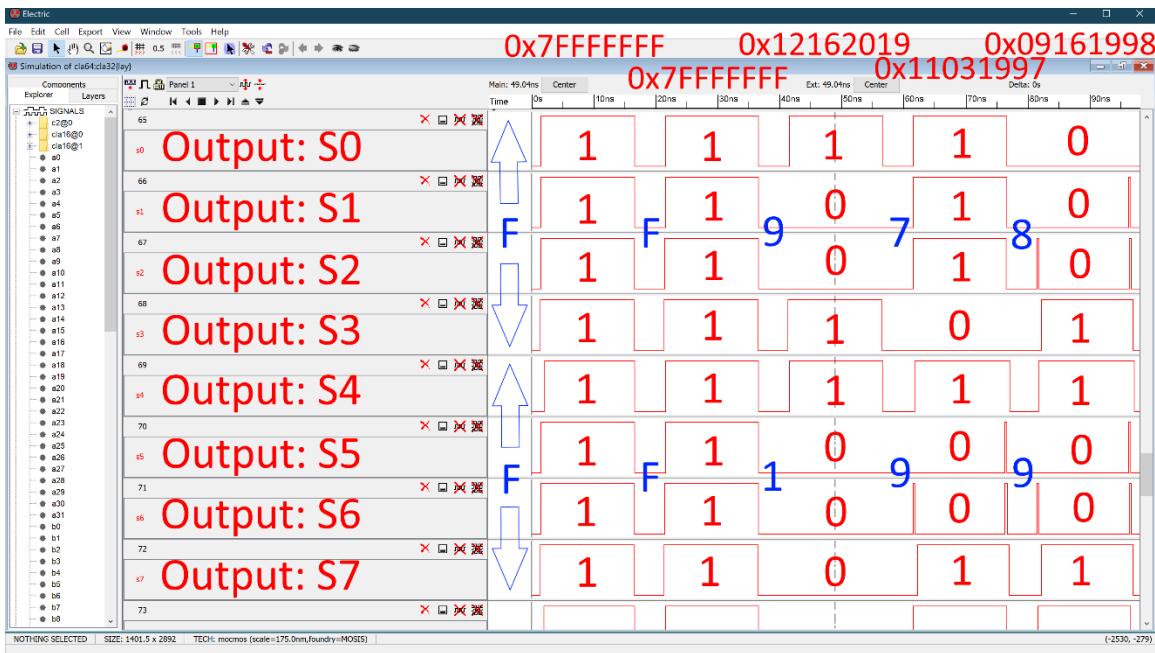


Figure 61: Subtraction Layout IRSIM - Output S[0:7]

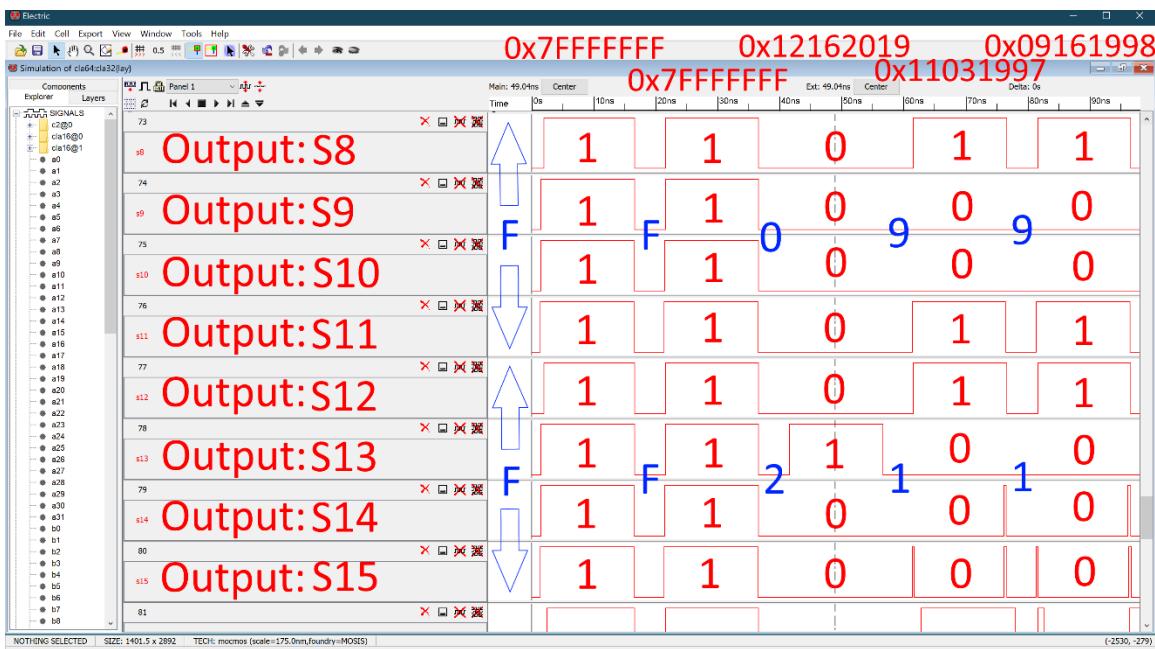
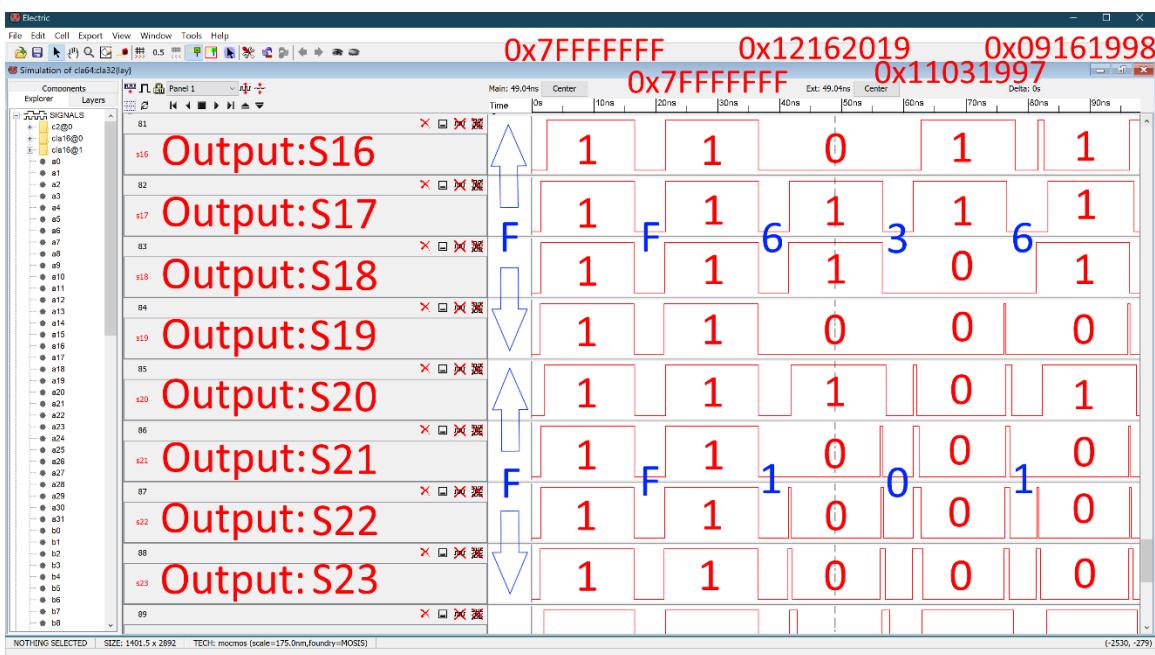
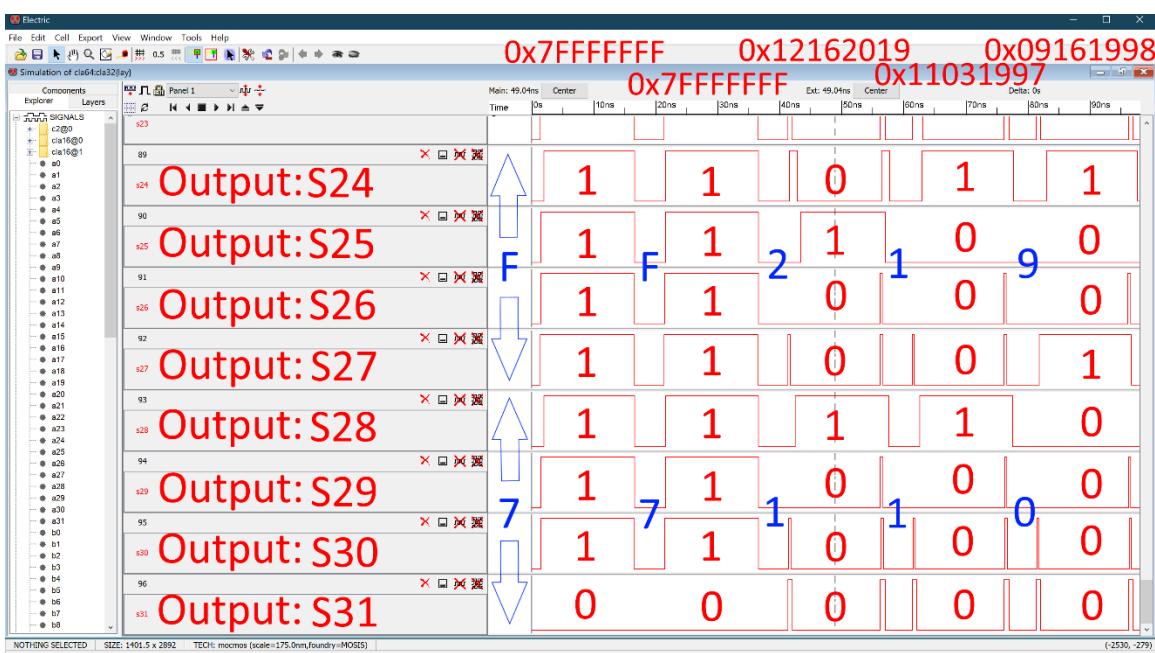


Figure 62: Subtraction Layout IRSIM - Output S[8:15]



**Figure 63: Subtraction Layout IRSIM - Output S[16:23]**



**Figure 64: Subtraction Layout IRSIM - Output S[23:31]**

Similarly to the schematic, the output matches with what we expected so we can confirm that the layout satisfies our design requirements. In addition, there are also delays in the layout as well.

### Section 5.3: Comparison

The results gathered from the IRSIM simulation for both schematic and layout are shown below. As we observed, the output for both the schematic and layout matches the expected output so it verifies that the logic of our circuit is correct. Before obtaining the correct output, spikes are seen in both the schematic and layout; that is due to the delay. When comparing the delay of the schematic and layout, the layout appears to have a larger delay. This is to be expected, due to the parasitic effects present in the layout.

**Table 2: Output Comparisons (IRSIM)**

	<b>Expected Output (Hexadecimal/Decimal)</b>	<b>Output Schematic (Hexadecimal/Decimal)</b>	<b>Output Layout (Hexadecimal/Decimal)</b>
<b>First Addition</b>	0x80000000	0x80000000	0x80000000
	2147483648	2147483648	2147483648
<b>Second Addition</b>	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
	4294967295	4294967295	4294967295
<b>Third Addition</b>	0x121A2019	0x121A2019	0x121A2019
	303702041	303702041	303702041
<b>Fourth Addition</b>	0x12162019	0x12162019	0x12162019
	303439897	303439897	303439897
<b>Fifth Addition</b>	0x12162019	0x12162019	0x12162019
	303439897	303439897	303439897
<b>First Subtraction</b>	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
	2147483647	2147483647	2147483647
<b>Second Subtraction</b>	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
	2147483647	2147483647	2147483647
<b>Third Subtraction</b>	0x12162019	0x12162019	0x12162019
<b>Fourth Subtraction</b>	303439897	303439897	303439897
<b>Fifth Subtraction</b>	0x11031997	0x11031997	0x11031997
	285415831	285415831	285415831
<b>Fifth Subtraction</b>	0x09161998	0x09161998	0x09161998
	152443288	152443288	152443288

## Section 6: LTSPICE Code and Parasitic Extractions

### Section 6.1: The SPICE Code

This section deals with the SPICE code that was used to measure the rise and fall times of all outputs. The code also measures the propagation delay at 50% voltage between input and output. 50% of V<sub>DD</sub> is 1.65 V, so our delay measurements will be at that value. Here, 32-bit input B will be at a constant value 0x00000000. Input A will also be at that value for a time. It will be pulsed to the value 0xFFFFFFFF for a time before returning to zero. The timing metrics will then be measured for each of the 32 bits of the S output.

As a side effect, the subtraction  $0 + (-1) = -1$  is also verified by this spice code, as all bits of output S will be pulsed to 1 and then back to zero. The waveforms are shown in the next section.

```
vdd vdd 0 DC 3.3
vgnd gnd 0 DC 0
va0 a0 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va1 a1 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va2 a2 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va3 a3 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va4 a4 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va5 a5 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va6 a6 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va7 a7 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va8 a8 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va9 a9 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va10 a10 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va11 a11 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va12 a12 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va13 a13 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va14 a14 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va15 a15 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va16 a16 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va17 a17 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va18 a18 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va19 a19 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va20 a20 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va21 a21 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va22 a22 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va23 a23 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va24 a24 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va25 a25 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va26 a26 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va27 a27 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va28 a28 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
```

```
va29 a29 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va30 a30 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
va31 a31 0 PULSE(3.3 0 250n 100p 100p 500n 1000n)
vb0 b0 0 DC 0
vb1 b1 0 DC 0
vb2 b2 0 DC 0
vb3 b3 0 DC 0
vb4 b4 0 DC 0
vb5 b5 0 DC 0
vb6 b6 0 DC 0
vb7 b7 0 DC 0
vb8 b8 0 DC 0
vb9 b9 0 DC 0
vb10 b10 0 DC 0
vb11 b11 0 DC 0
vb12 b12 0 DC 0
vb13 b13 0 DC 0
vb14 b14 0 DC 0
vb15 b15 0 DC 0
vb16 b16 0 DC 0
vb17 b17 0 DC 0
vb18 b18 0 DC 0
vb19 b19 0 DC 0
vb20 b20 0 DC 0
vb21 b21 0 DC 0
vb22 b22 0 DC 0
vb23 b23 0 DC 0
vb24 b24 0 DC 0
vb25 b25 0 DC 0
vb26 b26 0 DC 0
vb27 b27 0 DC 0
vb28 b28 0 DC 0
vb29 b29 0 DC 0
vb30 b30 0 DC 0
vb31 b31 0 DC 0
.meas TRAN tr0 TRIG V(s0) VAL=.33 RISE=1 TARG V(s0) VAL=2.97
RISE=1
.meas TRAN tr1 TRIG V(s1) VAL=.33 RISE=1 TARG V(s1) VAL=2.97
RISE=1
.meas TRAN tr2 TRIG V(s2) VAL=.33 RISE=1 TARG V(s2) VAL=2.97
RISE=1
.meas TRAN tr3 TRIG V(s3) VAL=.33 RISE=1 TARG V(s3) VAL=2.97
RISE=1
.meas TRAN tr4 TRIG V(s4) VAL=.33 RISE=1 TARG V(s4) VAL=2.97
RISE=1
```

```

.meas TRAN tr5 TRIG V(s5) VAL=.33 RISE=1 TARG V(s5) VAL=2.97
RISE=1
.meas TRAN tr6 TRIG V(s6) VAL=.33 RISE=1 TARG V(s6) VAL=2.97
RISE=1
.meas TRAN tr7 TRIG V(s7) VAL=.33 RISE=1 TARG V(s7) VAL=2.97
RISE=1
.meas TRAN tr8 TRIG V(s8) VAL=.33 RISE=1 TARG V(s8) VAL=2.97
RISE=1
.meas TRAN tr9 TRIG V(s9) VAL=.33 RISE=1 TARG V(s9) VAL=2.97
RISE=1
.meas TRAN tr10 TRIG V(s10) VAL=.33 RISE=1 TARG V(s10) VAL=2.97
RISE=1
.meas TRAN tr11 TRIG V(s11) VAL=.33 RISE=1 TARG V(s11) VAL=2.97
RISE=1
.meas TRAN tr12 TRIG V(s12) VAL=.33 RISE=1 TARG V(s12) VAL=2.97
RISE=1
.meas TRAN tr13 TRIG V(s13) VAL=.33 RISE=1 TARG V(s13) VAL=2.97
RISE=1
.meas TRAN tr14 TRIG V(s14) VAL=.33 RISE=1 TARG V(s14) VAL=2.97
RISE=1
.meas TRAN tr15 TRIG V(s15) VAL=.33 RISE=1 TARG V(s15) VAL=2.97
RISE=1
.meas TRAN tr16 TRIG V(s16) VAL=.33 RISE=1 TARG V(s16) VAL=2.97
RISE=1
.meas TRAN tr17 TRIG V(s17) VAL=.33 RISE=1 TARG V(s17) VAL=2.97
RISE=1
.meas TRAN tr18 TRIG V(s18) VAL=.33 RISE=1 TARG V(s18) VAL=2.97
RISE=1
.meas TRAN tr19 TRIG V(s19) VAL=.33 RISE=1 TARG V(s19) VAL=2.97
RISE=1
.meas TRAN tr20 TRIG V(s20) VAL=.33 RISE=1 TARG V(s20) VAL=2.97
RISE=1
.meas TRAN tr21 TRIG V(s21) VAL=.33 RISE=1 TARG V(s21) VAL=2.97
RISE=1
.meas TRAN tr22 TRIG V(s22) VAL=.33 RISE=1 TARG V(s22) VAL=2.97
RISE=1
.meas TRAN tr23 TRIG V(s23) VAL=.33 RISE=1 TARG V(s23) VAL=2.97
RISE=1
.meas TRAN tr24 TRIG V(s24) VAL=.33 RISE=1 TARG V(s24) VAL=2.97
RISE=1
.meas TRAN tr25 TRIG V(s25) VAL=.33 RISE=1 TARG V(s25) VAL=2.97
RISE=1
.meas TRAN tr26 TRIG V(s26) VAL=.33 RISE=1 TARG V(s26) VAL=2.97
RISE=1
.meas TRAN tr27 TRIG V(s27) VAL=.33 RISE=1 TARG V(s27) VAL=2.97

```

```

RISE=1
.meas TRAN tr28 TRIG V(s28) VAL=.33 RISE=1 TARG V(s28) VAL=2.97
RISE=1
.meas TRAN tr29 TRIG V(s29) VAL=.33 RISE=1 TARG V(s29) VAL=2.97
RISE=1
.meas TRAN tr30 TRIG V(s30) VAL=.33 RISE=1 TARG V(s30) VAL=2.97
RISE=1
.meas TRAN tr31 TRIG V(s31) VAL=.33 RISE=1 TARG V(s31) VAL=2.97
RISE=1
.meas TRAN tf0 TRIG V(s0) VAL=2.97 FALL=1 TARG V(s0) VAL=.33
FALL=1
.meas TRAN tf1 TRIG V(s1) VAL=2.97 FALL=1 TARG V(s1) VAL=.33
FALL=1
.meas TRAN tf2 TRIG V(s2) VAL=2.97 FALL=1 TARG V(s2) VAL=.33
FALL=1
.meas TRAN tf3 TRIG V(s3) VAL=2.97 FALL=1 TARG V(s3) VAL=.33
FALL=1
.meas TRAN tf4 TRIG V(s4) VAL=2.97 FALL=1 TARG V(s4) VAL=.33
FALL=1
.meas TRAN tf5 TRIG V(s5) VAL=2.97 FALL=1 TARG V(s5) VAL=.33
FALL=1
.meas TRAN tf6 TRIG V(s6) VAL=2.97 FALL=1 TARG V(s6) VAL=.33
FALL=1
.meas TRAN tf7 TRIG V(s7) VAL=2.97 FALL=1 TARG V(s7) VAL=.33
FALL=1
.meas TRAN tf8 TRIG V(s8) VAL=2.97 FALL=1 TARG V(s8) VAL=.33
FALL=1
.meas TRAN tf9 TRIG V(s9) VAL=2.97 FALL=1 TARG V(s9) VAL=.33
FALL=1
.meas TRAN tf10 TRIG V(s10) VAL=2.97 FALL=1 TARG V(s10) VAL=.33
FALL=1
.meas TRAN tf11 TRIG V(s11) VAL=2.97 FALL=1 TARG V(s11) VAL=.33
FALL=1
.meas TRAN tf12 TRIG V(s12) VAL=2.97 FALL=1 TARG V(s12) VAL=.33
FALL=1
.meas TRAN tf13 TRIG V(s13) VAL=2.97 FALL=1 TARG V(s13) VAL=.33
FALL=1
.meas TRAN tf14 TRIG V(s14) VAL=2.97 FALL=1 TARG V(s14) VAL=.33
FALL=1
.meas TRAN tf15 TRIG V(s15) VAL=2.97 FALL=1 TARG V(s15) VAL=.33
FALL=1
.meas TRAN tf16 TRIG V(s16) VAL=2.97 FALL=1 TARG V(s16) VAL=.33
FALL=1
.meas TRAN tf17 TRIG V(s17) VAL=2.97 FALL=1 TARG V(s17) VAL=.33
FALL=1

```

```

.meas TRAN tf18 TRIG V(s18) VAL=2.97 FALL=1 TARG V(s18) VAL=.33
FALL=1
.meas TRAN tf19 TRIG V(s19) VAL=2.97 FALL=1 TARG V(s19) VAL=.33
FALL=1
.meas TRAN tf20 TRIG V(s20) VAL=2.97 FALL=1 TARG V(s20) VAL=.33
FALL=1
.meas TRAN tf21 TRIG V(s21) VAL=2.97 FALL=1 TARG V(s21) VAL=.33
FALL=1
.meas TRAN tf22 TRIG V(s22) VAL=2.97 FALL=1 TARG V(s22) VAL=.33
FALL=1
.meas TRAN tf23 TRIG V(s23) VAL=2.97 FALL=1 TARG V(s23) VAL=.33
FALL=1
.meas TRAN tf24 TRIG V(s24) VAL=2.97 FALL=1 TARG V(s24) VAL=.33
FALL=1
.meas TRAN tf25 TRIG V(s25) VAL=2.97 FALL=1 TARG V(s25) VAL=.33
FALL=1
.meas TRAN tf26 TRIG V(s26) VAL=2.97 FALL=1 TARG V(s26) VAL=.33
FALL=1
.meas TRAN tf27 TRIG V(s27) VAL=2.97 FALL=1 TARG V(s27) VAL=.33
FALL=1
.meas TRAN tf28 TRIG V(s28) VAL=2.97 FALL=1 TARG V(s28) VAL=.33
FALL=1
.meas TRAN tf29 TRIG V(s29) VAL=2.97 FALL=1 TARG V(s29) VAL=.33
FALL=1
.meas TRAN tf30 TRIG V(s30) VAL=2.97 FALL=1 TARG V(s30) VAL=.33
FALL=1
.meas TRAN tf31 TRIG V(s31) VAL=2.97 FALL=1 TARG V(s31) VAL=.33
FALL=1
.meas TRAN tplh0 TRIG V(a0) VAL=1.65 RISE=1 TARG V(s0) VAL=1.65
RISE=1
.meas TRAN tplh1 TRIG V(a1) VAL=1.65 RISE=1 TARG V(s1) VAL=1.65
RISE=1
.meas TRAN tplh2 TRIG V(a2) VAL=1.65 RISE=1 TARG V(s2) VAL=1.65
RISE=1
.meas TRAN tplh3 TRIG V(a3) VAL=1.65 RISE=1 TARG V(s3) VAL=1.65
RISE=1
.meas TRAN tplh4 TRIG V(a4) VAL=1.65 RISE=1 TARG V(s4) VAL=1.65
RISE=1
.meas TRAN tplh5 TRIG V(a5) VAL=1.65 RISE=1 TARG V(s5) VAL=1.65
RISE=1
.meas TRAN tplh6 TRIG V(a6) VAL=1.65 RISE=1 TARG V(s6) VAL=1.65
RISE=1
.meas TRAN tplh7 TRIG V(a7) VAL=1.65 RISE=1 TARG V(s7) VAL=1.65
RISE=1
.meas TRAN tplh8 TRIG V(a8) VAL=1.65 RISE=1 TARG V(s8) VAL=1.65

```

```

RISE=1
.meas TRAN tplh9 TRIG V(a9) VAL=1.65 RISE=1 TARG V(s9) VAL=1.65
RISE=1
.meas TRAN tplh10 TRIG V(a10) VAL=1.65 RISE=1 TARG V(s10)
VAL=1.65 RISE=1
.meas TRAN tplh11 TRIG V(a11) VAL=1.65 RISE=1 TARG V(s11)
VAL=1.65 RISE=1
.meas TRAN tplh12 TRIG V(a12) VAL=1.65 RISE=1 TARG V(s12)
VAL=1.65 RISE=1
.meas TRAN tplh13 TRIG V(a13) VAL=1.65 RISE=1 TARG V(s13)
VAL=1.65 RISE=1
.meas TRAN tplh14 TRIG V(a14) VAL=1.65 RISE=1 TARG V(s14)
VAL=1.65 RISE=1
.meas TRAN tplh15 TRIG V(a15) VAL=1.65 RISE=1 TARG V(s15)
VAL=1.65 RISE=1
.meas TRAN tplh16 TRIG V(a16) VAL=1.65 RISE=1 TARG V(s16)
VAL=1.65 RISE=1
.meas TRAN tplh17 TRIG V(a17) VAL=1.65 RISE=1 TARG V(s17)
VAL=1.65 RISE=1
.meas TRAN tplh18 TRIG V(a18) VAL=1.65 RISE=1 TARG V(s18)
VAL=1.65 RISE=1
.meas TRAN tplh19 TRIG V(a19) VAL=1.65 RISE=1 TARG V(s19)
VAL=1.65 RISE=1
.meas TRAN tplh20 TRIG V(a20) VAL=1.65 RISE=1 TARG V(s20)
VAL=1.65 RISE=1
.meas TRAN tplh21 TRIG V(a21) VAL=1.65 RISE=1 TARG V(s21)
VAL=1.65 RISE=1
.meas TRAN tplh22 TRIG V(a22) VAL=1.65 RISE=1 TARG V(s22)
VAL=1.65 RISE=1
.meas TRAN tplh23 TRIG V(a23) VAL=1.65 RISE=1 TARG V(s23)
VAL=1.65 RISE=1
.meas TRAN tplh24 TRIG V(a24) VAL=1.65 RISE=1 TARG V(s24)
VAL=1.65 RISE=1
.meas TRAN tplh25 TRIG V(a25) VAL=1.65 RISE=1 TARG V(s25)
VAL=1.65 RISE=1
.meas TRAN tplh26 TRIG V(a26) VAL=1.65 RISE=1 TARG V(s26)
VAL=1.65 RISE=1
.meas TRAN tplh27 TRIG V(a27) VAL=1.65 RISE=1 TARG V(s27)
VAL=1.65 RISE=1
.meas TRAN tplh28 TRIG V(a28) VAL=1.65 RISE=1 TARG V(s28)
VAL=1.65 RISE=1
.meas TRAN tplh29 TRIG V(a29) VAL=1.65 RISE=1 TARG V(s29)
VAL=1.65 RISE=1
.meas TRAN tplh30 TRIG V(a30) VAL=1.65 RISE=1 TARG V(s30)
VAL=1.65 RISE=1

```

```

.meas TRAN tplh31 TRIG V(a31) VAL=1.65 RISE=1 TARG V(s31)
VAL=1.65 RISE=1
.meas TRAN tphl0 TRIG V(a0) VAL=1.65 FALL=1 TARG V(s0) VAL=1.65
FALL=1
.meas TRAN tphl1 TRIG V(a1) VAL=1.65 FALL=1 TARG V(s1) VAL=1.65
FALL=1
.meas TRAN tphl2 TRIG V(a2) VAL=1.65 FALL=1 TARG V(s2) VAL=1.65
FALL=1
.meas TRAN tphl3 TRIG V(a3) VAL=1.65 FALL=1 TARG V(s3) VAL=1.65
FALL=1
.meas TRAN tphl4 TRIG V(a4) VAL=1.65 FALL=1 TARG V(s4) VAL=1.65
FALL=1
.meas TRAN tphl5 TRIG V(a5) VAL=1.65 FALL=1 TARG V(s5) VAL=1.65
FALL=1
.meas TRAN tphl6 TRIG V(a6) VAL=1.65 FALL=1 TARG V(s6) VAL=1.65
FALL=1
.meas TRAN tphl7 TRIG V(a7) VAL=1.65 FALL=1 TARG V(s7) VAL=1.65
FALL=1
.meas TRAN tphl8 TRIG V(a8) VAL=1.65 FALL=1 TARG V(s8) VAL=1.65
FALL=1
.meas TRAN tphl9 TRIG V(a9) VAL=1.65 FALL=1 TARG V(s9) VAL=1.65
FALL=1
.meas TRAN tphl10 TRIG V(a10) VAL=1.65 FALL=1 TARG V(s10)
VAL=1.65 FALL=1
.meas TRAN tphl11 TRIG V(a11) VAL=1.65 FALL=1 TARG V(s11)
VAL=1.65 FALL=1
.meas TRAN tphl12 TRIG V(a12) VAL=1.65 FALL=1 TARG V(s12)
VAL=1.65 FALL=1
.meas TRAN tphl13 TRIG V(a13) VAL=1.65 FALL=1 TARG V(s13)
VAL=1.65 FALL=1
.meas TRAN tphl14 TRIG V(a14) VAL=1.65 FALL=1 TARG V(s14)
VAL=1.65 FALL=1
.meas TRAN tphl15 TRIG V(a15) VAL=1.65 FALL=1 TARG V(s15)
VAL=1.65 FALL=1
.meas TRAN tphl16 TRIG V(a16) VAL=1.65 FALL=1 TARG V(s16)
VAL=1.65 FALL=1
.meas TRAN tphl17 TRIG V(a17) VAL=1.65 FALL=1 TARG V(s17)
VAL=1.65 FALL=1
.meas TRAN tphl18 TRIG V(a18) VAL=1.65 FALL=1 TARG V(s18)
VAL=1.65 FALL=1
.meas TRAN tphl19 TRIG V(a19) VAL=1.65 FALL=1 TARG V(s19)
VAL=1.65 FALL=1
.meas TRAN tphl20 TRIG V(a20) VAL=1.65 FALL=1 TARG V(s20)
VAL=1.65 FALL=1
.meas TRAN tphl21 TRIG V(a21) VAL=1.65 FALL=1 TARG V(s21)

```

```
VAL=1.65 FALL=1
.meas TRAN tph122 TRIG V(a22) VAL=1.65 FALL=1 TARG V(s22)
VAL=1.65 FALL=1
.meas TRAN tph123 TRIG V(a23) VAL=1.65 FALL=1 TARG V(s23)
VAL=1.65 FALL=1
.meas TRAN tph124 TRIG V(a24) VAL=1.65 FALL=1 TARG V(s24)
VAL=1.65 FALL=1
.meas TRAN tph125 TRIG V(a25) VAL=1.65 FALL=1 TARG V(s25)
VAL=1.65 FALL=1
.meas TRAN tph126 TRIG V(a26) VAL=1.65 FALL=1 TARG V(s26)
VAL=1.65 FALL=1
.meas TRAN tph127 TRIG V(a27) VAL=1.65 FALL=1 TARG V(s27)
VAL=1.65 FALL=1
.meas TRAN tph128 TRIG V(a28) VAL=1.65 FALL=1 TARG V(s28)
VAL=1.65 FALL=1
.meas TRAN tph129 TRIG V(a29) VAL=1.65 FALL=1 TARG V(s29)
VAL=1.65 FALL=1
.meas TRAN tph130 TRIG V(a30) VAL=1.65 FALL=1 TARG V(s30)
VAL=1.65 FALL=1
.meas TRAN tph131 TRIG V(a31) VAL=1.65 FALL=1 TARG V(s31)
.tran 1000n
.include C:\Electric\kim_models.txt
```

Figure 65: SPICE Code That Measures Rise Time, Fall Time, and Propagation Delay

## Section 6.2: Parasitic Analysis

After finishing the 32-bit Carry Look-Ahead Adder layout, LTSPICE produced the following parasitic RC extraction netlist for the top level cell.

```
*** TOP LEVEL CELL: cla32{lay}
Xc2@0    net@17#2contact@9_metal-2-metal-3      gnd      net@1      gnd
net@9#1contact@2_metal-1-metal-2 vdd cla64_c1
Xcla16@0 a16 a17 a26 a27 a28 a29 a30 a31 a18 a19 a20 a21 a22 a23
a24 a25 b16 b17 b26 b27 b28 b29 b30 b31 b18 b19 b20 b21 b22 b23
b24 b25 net@17 gnd gnd gnd s16 s17 s26 s27 s28 s29 s30 s31 s18
s19 s20 s21 s22 s23 s24 s25 vdd cla64_cla16
Xcla16@1 a0 a1 a10 a11 a12 a13 a14 a15 a2 a3 a4 a5 a6 a7 a8 a9 b0
b1 b10 b11 b12 b13 b14 b15 b2 b3 b4 b5 b6 b7 b8 b9 gnd net@1 gnd
net@9 s0 s1 s10 s11 s12 s13 s14 s15 s2 s3 s4 s5 s6 s7 s8 s9 vdd
cla64_cla16
** Extracted Parasitic Capacitors ***
C0 net@17#2contact@9_metal-2-metal-3 0 18.714fF
C1 net@1 0 6.078fF
C2 net@9#1contact@2_metal-1-metal-2 0 4.453fF
C3 net@17 0 17.627fF
C4 s16 0 7.288fF
C5 s17 0 6.129fF
C6 s18 0 6.129fF
C7 s19 0 6.129fF
C8 s20 0 6.129fF
C9 s21 0 6.129fF
C10 s22 0 6.129fF
C11 s23 0 6.129fF
C12 s24 0 6.129fF
C13 s25 0 6.129fF
C14 s26 0 6.129fF
C15 s27 0 6.129fF
C16 s28 0 6.129fF
C17 s29 0 6.129fF
C18 s30 0 6.129fF
C19 s31 0 6.129fF
C20 net@9 0 4.951fF
C21 s0 0 6.129fF
C22 s1 0 6.129fF
C23 s2 0 6.37fF
C24 s3 0 6.37fF
C25 s4 0 6.37fF
C26 s5 0 6.37fF
C27 s6 0 6.37fF
C28 s7 0 6.37fF
```

```
C29 s8 0 6.37fF
C30 s9 0 6.37fF
C31 s10 0 6.37fF
C32 s11 0 6.37fF
C33 s12 0 6.37fF
C34 s13 0 6.37fF
C35 s14 0 6.37fF
C36 s15 0 6.37fF
** Extracted Parasitic Resistors ***
R0 net@9 net@9#1contact@2_metal-1-metal-2 4.979
R1 net@17#2contact@9_metal-2-metal-3 net@17#2contact@9_metal-2-
metal-3#0 6.921
C37 net@17#2contact@9_metal-2-metal-3##0 0 15.639fF
R2 net@17#2contact@9_metal-2-metal-3##0 net@17#2contact@9_metal-
2-metal-3##1 6.921
C38 net@17#2contact@9_metal-2-metal-3##1 0 15.639fF
R3 net@17#2contact@9_metal-2-metal-3##1 net@17 6.921
```

Figure 66: Extracted Parasitic RC for Top-Level 32-bit CLA Circuit

For convenience, all 32 output parasitic capacitors are listed in the table below.

**Table 3: Output Parasitic Capacitances of 32-bit CLA**

Output	Capacitance
s0	6.129fF
s1	6.129fF
s2	6.37fF
s3	6.37fF
s4	6.37fF
s5	6.37fF
s6	6.37fF
s7	6.37fF
s8	6.37fF
s9	6.37fF
s10	6.37fF
s11	6.37fF
s12	6.37fF
s13	6.37fF
s14	6.37fF
s15	6.37fF
s16	7.288fF
s17	6.129fF
s18	6.129fF
s19	6.129fF
s20	6.129fF
s21	6.129fF
s22	6.129fF
s23	6.129fF
s24	6.129fF
s25	6.129fF
s26	6.129fF
s27	6.129fF
s28	6.129fF
s29	6.129fF
s30	6.129fF
s31	6.129fF

We then examine the extracted RC parasitic analysis generated by LTSPICE for the sub-components. Specifically, we will be analyzing the parasitic resistors present in the layouts for the CMOS Inverter, the XOR Gate, the NAND Gate, and the PG4 module, which functions as a 4-input AND gate.

Below are some of the LTSPICE-generated parasitic netlists of those components.

```
*** SUBCIRCUIT cla64_inv_well FROM CELL inv_well{lay}
.SUBCKT cla64_inv_well gnd in out vdd
Mnmos@0 out in#3nmos@0_poly-right gnd gnd N L=0.35U W=0.875U
AS=3.752P AD=1.263P PS=12.075U PD=4.55U
Mpmos@0 out in#1pmos@0_poly-left vdd vdd P L=0.35U W=1.75U
AS=5.176P AD=1.263P PS=14.175U PD=4.55U
** Extracted Parasitic Capacitors ***
C0 out 0 3.234fF
C1 in 0 0.313fF
C2 in#2pin@0_polysilicon-1 0 0.201fF
** Extracted Parasitic Resistors ***
R0 in#1pmos@0_poly-left in#1pmos@0_poly-left##0 9.455
R1 in#1pmos@0_poly-left##0 in#1pmos@0_poly-left##1 9.455
R2 in#1pmos@0_poly-left##1 in#1pmos@0_poly-left##2 9.455
R3 in#1pmos@0_poly-left##2 in#1pmos@0_poly-left##3 9.455
R4 in#1pmos@0_poly-left##3 in#1pmos@0_poly-left##4 9.455
R5 in#1pmos@0_poly-left##4 in#1pmos@0_poly-left##5 9.455
R6 in#1pmos@0_poly-left##5 in#1pmos@0_poly-left##6 9.455
R7 in#1pmos@0_poly-left##6 in#1pmos@0_poly-left##7 9.455
R8 in#1pmos@0_poly-left##7 in#1pmos@0_poly-left##8 9.455
R9 in#1pmos@0_poly-left##8 in#2pin@0_polysilicon-1 9.455
R10 in#2pin@0_polysilicon-1 in#2pin@0_polysilicon-1##0 8.525
R11 in#2pin@0_polysilicon-1##0 in#2pin@0_polysilicon-1##1 8.525
R12 in#2pin@0_polysilicon-1##1 in#2pin@0_polysilicon-1##2 8.525
R13 in#2pin@0_polysilicon-1##2 in#2pin@0_polysilicon-1##3 8.525
R14 in#2pin@0_polysilicon-1##3 in#2pin@0_polysilicon-1##4 8.525
R15 in#2pin@0_polysilicon-1##4 in#3nmos@0_poly-right 8.525
R16 in#2pin@0_polysilicon-1 in 9.3
.ENDS cla64_inv_well

*** SUBCIRCUIT cla64_inv FROM CELL inv{lay}
.SUBCKT cla64_inv gnd in out vdd
Mnmos@1 out in#2nmos@1_poly-right gnd gnd N L=0.35U W=0.875U
AS=0.842P AD=1.263P PS=3.675U PD=4.55U
Mpmos@1 out in#0pmos@1_poly-left vdd vdd P L=0.35U W=1.75U
AS=1.684P AD=1.263P PS=5.425U PD=4.55U
** Extracted Parasitic Capacitors ***
C0 out 0 1.592fF
C1 in 0 0.313fF
C2 in#1pin@3_polysilicon-1 0 0.169fF
** Extracted Parasitic Resistors ***
R0 in#0pmos@1_poly-left in#0pmos@1_poly-left##0 7.75
```

```

R1 in#0pmos@1_poly-left##0 in#0pmos@1_poly-left##1 7.75
R2 in#0pmos@1_poly-left##1 in#0pmos@1_poly-left##2 7.75
R3 in#0pmos@1_poly-left##2 in#1pin@3_polysilicon-1 7.75
R4 in#1pin@3_polysilicon-1 in#1pin@3_polysilicon-1##0 8.138
R5 in#1pin@3_polysilicon-1##0 in#1pin@3_polysilicon-1##1 8.138
R6 in#1pin@3_polysilicon-1##1 in#1pin@3_polysilicon-1##2 8.138
R7 in#1pin@3_polysilicon-1##2 in#2nmos@1_poly-right 8.138
R8 in#1pin@3_polysilicon-1 in 9.3
.ENDS cla64_inv

*** SUBCIRCUIT cla64_nand FROM CELL nand{lay}
.SUBCKT cla64_nand A B gnd O vdd
Mnmos@3 O A#2nmos@3_poly-right net@134 gnd N L=0.35U W=0.875U
AS=0.459P AD=0.893P PS=1.925U PD=3.092U
Mnmos@4 net@134 B#2nmos@4_poly-right gnd gnd N L=0.35U W=0.875U
AS=4.977P AD=0.459P PS=14.875U PD=1.925U
Mpmos@3 vdd A#0pmos@3_poly-left O vdd P L=0.35U W=1.75U AS=0.893P
AD=4.165P PS=3.092U PD=11.2U
Mpmos@4 O B#0pmos@4_poly-left vdd vdd P L=0.35U W=1.75U AS=4.165P
AD=0.893P PS=11.2U PD=3.092U
** Extracted Parasitic Capacitors ***
C0 O 0 4.079fF
C1 B 0 0.293fF
C2 A 0 1.123fF
C3 B#1pin@3_polysilicon-1 0 0.212fF
** Extracted Parasitic Resistors ***
R0 B#0pmos@4_poly-left B#0pmos@4_poly-left##0 9.688
R1 B#0pmos@4_poly-left##0 B#0pmos@4_poly-left##1 9.688
R2 B#0pmos@4_poly-left##1 B#0pmos@4_poly-left##2 9.688
R3 B#0pmos@4_poly-left##2 B#0pmos@4_poly-left##3 9.688
R4 B#0pmos@4_poly-left##3 B#0pmos@4_poly-left##4 9.688
R5 B#0pmos@4_poly-left##4 B#0pmos@4_poly-left##5 9.688
R6 B#0pmos@4_poly-left##5 B#0pmos@4_poly-left##6 9.688
R7 B#0pmos@4_poly-left##6 B#0pmos@4_poly-left##7 9.688
R8 B#0pmos@4_poly-left##7 B#0pmos@4_poly-left##8 9.688
R9 B#0pmos@4_poly-left##8 B#0pmos@4_poly-left##9 9.688
R10 B#0pmos@4_poly-left##9 B#0pmos@4_poly-left##10 9.688
R11 B#0pmos@4_poly-left##10 B#1pin@3_polysilicon-1 9.688
R12 B#1pin@3_polysilicon-1 B#1pin@3_polysilicon-1##0 9.817
R13 B#1pin@3_polysilicon-1##0 B#1pin@3_polysilicon-1##1 9.817
R14 B#1pin@3_polysilicon-1##1 B#2nmos@4_poly-right 9.817
R15 B#1pin@3_polysilicon-1 B#1pin@3_polysilicon-1##0 7.233
R16 B#1pin@3_polysilicon-1##0 B#1pin@3_polysilicon-1##1 7.233
R17 B#1pin@3_polysilicon-1##1 B 7.233

```

```

R18 A#0pmos@3_poly-left A#0pmos@3_poly-left##0 9.854
R19 A#0pmos@3_poly-left##0 A#0pmos@3_poly-left##1 9.854
R20 A#0pmos@3_poly-left##1 A#0pmos@3_poly-left##2 9.854
R21 A#0pmos@3_poly-left##2 A#0pmos@3_poly-left##3 9.854
R22 A#0pmos@3_poly-left##3 A#0pmos@3_poly-left##4 9.854
R23 A#0pmos@3_poly-left##4 A#0pmos@3_poly-left##5 9.854
R24 A#0pmos@3_poly-left##5 A#0pmos@3_poly-left##6 9.854
R25 A#0pmos@3_poly-left##6 A#0pmos@3_poly-left##7 9.854
R26 A#0pmos@3_poly-left##7 A#0pmos@3_poly-left##8 9.854
R27 A#0pmos@3_poly-left##8 A#0pmos@3_poly-left##9 9.854
R28 A#0pmos@3_poly-left##9 A#0pmos@3_poly-left##10 9.854
R29 A#0pmos@3_poly-left##10 A#0pmos@3_poly-left##11 9.854
R30 A#0pmos@3_poly-left##11 A#0pmos@3_poly-left##12 9.854
R31 A#0pmos@3_poly-left##12 A 9.854
R32 A#2nmos@3_poly-right A 7.75
.ENDS cla64_nand

*** SUBCIRCUIT cla64_xor FROM CELL xor{lay}
.SUBCKT cla64_xor A B gnd O vdd
Xinv@0 gnd A net@51#1contact@14_metal-1-polysilicon-1 vdd
cla64_inv
Xinv@1 gnd B net@68#1contact@15_metal-1-polysilicon-1 vdd
cla64_inv
Mnmos@0 O net@51#2nmos@0_poly-right net@2 gnd N L=0.35U W=0.875U
AS=0.842P AD=0.689P PS=3.675U PD=2.363U
Mnmos@1 net@3 A#0nmos@1_poly-right O gnd N L=0.35U W=0.875U
AS=0.689P AD=0.459P PS=2.363U PD=1.925U
Mnmos@2 gnd B#1nmos@2_poly-right net@3 gnd N L=0.35U W=0.875U
AS=0.459P AD=8.116P PS=1.925U PD=20.3U
Mnmos@3 net@2 net@68#2nmos@3_poly-right gnd gnd N L=0.35U
W=0.875U AS=8.116P AD=0.842P PS=20.3U PD=3.675U
Mppmos@0 net@17 net@51 vdd vdd P L=0.35U W=1.75U AS=10.872P
AD=0.919P PS=23.975U PD=2.8U
Mppmos@1 O A#1pmos@1_poly-left net@17 vdd P L=0.35U W=1.75U
AS=0.919P AD=0.689P PS=2.8U PD=2.363U
Mppmos@2 net@17 B#0pmos@2_poly-left O vdd P L=0.35U W=1.75U
AS=0.689P AD=0.919P PS=2.363U PD=2.8U
Mppmos@3 vdd net@68 net@17 vdd P L=0.35U W=1.75U AS=0.919P
AD=10.872P PS=2.8U PD=23.975U
** Extracted Parasitic Capacitors ***
C0 O 0 4.765fF
C1 net@2 0 2.135fF
C2 net@17 0 1.664fF
C3 net@51#1contact@14_metal-1-polysilicon-1 0 0.92fF

```

```

C4 net@68#1contact@15_metal-1-polysilicon-1 0 0.92fF
C5 A 0 3.339fF
C6 B 0 7.934fF
C7 A#1pmos@1_poly-left 0 0.115fF
C8 B#0pmos@2_poly-left 0 0.115fF
** Extracted Parasitic Resistors ***
R0 net@51 net@51##0 7.75
R1 net@51##0 net@51##1 7.75
R2 net@51##1 net@51##2 7.75
R3 net@51##2 net@51#1contact@14_metal-1-polysilicon-1 7.75
R4 net@51#1contact@14_metal-1-polysilicon-1
net@51#1contact@14_metal-1-polysilicon-1##0 8.138
R5 net@51#1contact@14_metal-1-polysilicon-1##0
net@51#1contact@14_metal-1-polysilicon-1##1 8.138
R6 net@51#1contact@14_metal-1-polysilicon-1##1
net@51#1contact@14_metal-1-polysilicon-1##2 8.138
R7 net@51#1contact@14_metal-1-polysilicon-1##2
net@51#2nmos@0_poly-right 8.138
R8 A#0nmos@1_poly-right A#0nmos@1_poly-right##0 9.079
R9 A#0nmos@1_poly-right##0 A#0nmos@1_poly-right##1 9.079
R10 A#0nmos@1_poly-right##1 A#0nmos@1_poly-right##2 9.079
R11 A#0nmos@1_poly-right##2 A#0nmos@1_poly-right##3 9.079
R12 A#0nmos@1_poly-right##3 A#0nmos@1_poly-right##4 9.079
R13 A#0nmos@1_poly-right##4 A#0nmos@1_poly-right##5 9.079
R14 A#0nmos@1_poly-right##5 A#1pmos@1_poly-left 9.079
R15 net@68 net@68##0 7.75
R16 net@68##0 net@68##1 7.75
R17 net@68##1 net@68##2 7.75
R18 net@68##2 net@68#1contact@15_metal-1-polysilicon-1 7.75
R19 net@68#1contact@15_metal-1-polysilicon-1
net@68#1contact@15_metal-1-polysilicon-1##0 8.138
R20 net@68#1contact@15_metal-1-polysilicon-1##0
net@68#1contact@15_metal-1-polysilicon-1##1 8.138
R21 net@68#1contact@15_metal-1-polysilicon-1##1
net@68#1contact@15_metal-1-polysilicon-1##2 8.138
R22 net@68#1contact@15_metal-1-polysilicon-1##2
net@68#2nmos@3_poly-right 8.138
R23 B#0pmos@2_poly-left B#0pmos@2_poly-left##0 9.079
R24 B#0pmos@2_poly-left##0 B#0pmos@2_poly-left##1 9.079
R25 B#0pmos@2_poly-left##1 B#0pmos@2_poly-left##2 9.079
R26 B#0pmos@2_poly-left##2 B#0pmos@2_poly-left##3 9.079
R27 B#0pmos@2_poly-left##3 B#0pmos@2_poly-left##4 9.079
R28 B#0pmos@2_poly-left##4 B#0pmos@2_poly-left##5 9.079
R29 B#0pmos@2_poly-left##5 B#1nmos@2_poly-right 9.079
R30 B#0pmos@2_poly-left B 7.75

```

```

R31 A#1pmos@1_poly-left A 7.75
.ENDS cla64_xor

*** SUBCIRCUIT cla64_pg4 FROM CELL pg4{lay}
.SUBCKT cla64_pg4 gnd p0 p1 p2 p3 pg vdd
Xinv_well@1 gnd net@0 pg vdd cla64_inv_well
Mnmos@0 net@98 p1#2nmox@0_poly-right net@97 gnd N L=0.35U
W=0.875U AS=0.459P AD=0.459P PS=1.925U PD=1.925U
Mnmos@1 net@97 p0#2nmox@1_poly-right net@0 gnd N L=0.35U W=0.875U
AS=0.903P AD=0.459P PS=2.975U PD=1.925U
Mnmos@2 net@99 p2#2nmox@2_poly-right net@98 gnd N L=0.35U
W=0.875U AS=0.459P AD=0.459P PS=1.925U PD=1.925U
Mnmos@3 gnd p3#2nmox@3_poly-right net@99 gnd N L=0.35U W=0.875U
AS=0.459P AD=7.656P PS=1.925U PD=21U
Mpmos@0 vdd p1#0pmos@0_poly-left net@0 vdd P L=0.35U W=1.75U
AS=0.903P AD=3.361P PS=2.975U PD=8.531U
Mpmos@1 net@0 p0#0pmos@1_poly-left vdd vdd P L=0.35U W=1.75U
AS=3.361P AD=0.903P PS=8.531U PD=2.975U
Mpmos@2 net@0 p2#0pmos@2_poly-left vdd vdd P L=0.35U W=1.75U
AS=3.361P AD=0.903P PS=8.531U PD=2.975U
Mpmos@3 vdd p3#0pmos@3_poly-left net@0 vdd P L=0.35U W=1.75U
AS=0.903P AD=3.361P PS=2.975U PD=8.531U
** Extracted Parasitic Capacitors ***
C0 net@0 0 5.842fF
C1 p0 0 0.662fF
C2 p1 0 1.086fF
C3 p2 0 1.476fF
C4 p3 0 1.853fF
** Extracted Parasitic Resistors ***
R0 p0#0pmos@1_poly-left p0#0pmos@1_poly-left##0 7.233
R1 p0#0pmos@1_poly-left##0 p0#0pmos@1_poly-left##1 7.233
R2 p0#0pmos@1_poly-left##1 p0 7.233
R3 p0 p0##0 9.538
R4 p0##0 p0##1 9.538
R5 p0##1 p0##2 9.538
R6 p0##2 p0##3 9.538
R7 p0##3 p0##4 9.538
R8 p0##4 p0##5 9.538
R9 p0##5 p0##6 9.538
R10 p0##6 p0##7 9.538
R11 p0##7 p0##8 9.538
R12 p0##8 p0##9 9.538
R13 p0##9 p0##10 9.538
R14 p0##10 p0##11 9.538

```

```

R15 p0##11 p0#2nmos@1_poly-right 9.538
R16 p1#0pmos@0_poly-left p1#0pmos@0_poly-left##0 9.3
R17 p1#0pmos@0_poly-left##0 p1#0pmos@0_poly-left##1 9.3
R18 p1#0pmos@0_poly-left##1 p1#0pmos@0_poly-left##2 9.3
R19 p1#0pmos@0_poly-left##2 p1#0pmos@0_poly-left##3 9.3
R20 p1#0pmos@0_poly-left##3 p1#0pmos@0_poly-left##4 9.3
R21 p1#0pmos@0_poly-left##4 p1 9.3
R22 p1 p1##0 9.989
R23 p1##0 p1##1 9.989
R24 p1##1 p1##2 9.989
R25 p1##2 p1##3 9.989
R26 p1##3 p1##4 9.989
R27 p1##4 p1##5 9.989
R28 p1##5 p1##6 9.989
R29 p1##6 p1##7 9.989
R30 p1##7 p1#2nmos@0_poly-right 9.989
R31 p2#0pmos@2_poly-left p2#0pmos@2_poly-left##0 9.128
R32 p2#0pmos@2_poly-left##0 p2#0pmos@2_poly-left##1 9.128
R33 p2#0pmos@2_poly-left##1 p2#0pmos@2_poly-left##2 9.128
R34 p2#0pmos@2_poly-left##2 p2#0pmos@2_poly-left##3 9.128
R35 p2#0pmos@2_poly-left##3 p2#0pmos@2_poly-left##4 9.128
R36 p2#0pmos@2_poly-left##4 p2#0pmos@2_poly-left##5 9.128
R37 p2#0pmos@2_poly-left##5 p2#0pmos@2_poly-left##6 9.128
R38 p2#0pmos@2_poly-left##6 p2#0pmos@2_poly-left##7 9.128
R39 p2#0pmos@2_poly-left##7 p2 9.128
R40 p2 p2##0 9.079
R41 p2##0 p2##1 9.079
R42 p2##1 p2##2 9.079
R43 p2##2 p2##3 9.079
R44 p2##3 p2##4 9.079
R45 p2##4 p2##5 9.079
R46 p2##5 p2#2nmos@2_poly-right 9.079
R47 p3#0pmos@3_poly-left p3#0pmos@3_poly-left##0 9.658
R48 p3#0pmos@3_poly-left##0 p3#0pmos@3_poly-left##1 9.658
R49 p3#0pmos@3_poly-left##1 p3#0pmos@3_poly-left##2 9.658
R50 p3#0pmos@3_poly-left##2 p3#0pmos@3_poly-left##3 9.658
R51 p3#0pmos@3_poly-left##3 p3#0pmos@3_poly-left##4 9.658
R52 p3#0pmos@3_poly-left##4 p3#0pmos@3_poly-left##5 9.658
R53 p3#0pmos@3_poly-left##5 p3#0pmos@3_poly-left##6 9.658
R54 p3#0pmos@3_poly-left##6 p3#0pmos@3_poly-left##7 9.658
R55 p3#0pmos@3_poly-left##7 p3#0pmos@3_poly-left##8 9.658
R56 p3#0pmos@3_poly-left##8 p3#0pmos@3_poly-left##9 9.658
R57 p3#0pmos@3_poly-left##9 p3#0pmos@3_poly-left##10 9.658
R58 p3#0pmos@3_poly-left##10 p3#0pmos@3_poly-left##11 9.658
R59 p3#0pmos@3_poly-left##11 p3 9.658

```

```
R60 p3 p3##0 6.717
R61 p3##0 p3##1 6.717
R62 p3##1 p3#2nmos@3_poly-right 6.717
.ENDS cla64_pg4
```

Figure 67: Extracted Parasitic RC for Lower-Level Components

Below is a table of the  $R_p$  and  $R_N$  values extracted from the parasitic analysis. All resistor units are in ohms. As mentioned earlier, the inverter and NAND gates also have slightly taller versions of their layouts to allow more room for wiring in between their PUNs and PDNs. Their values are also recorded here.

**Table 4: Parasitic  $R_p$  and  $R_N$  extracted from Fundamental Building Blocks**

Component	$R_p$ ( $\Omega$ )	$R_N$ ( $\Omega$ )
<b>Inverter (Tall)</b>	8.138	7.75
<b>Inverter</b>	9.455	8.525
<b>Inverter (no wells)</b>	9.455	8.525
<b>NAND (A input)</b>	9.854	7.75
<b>NAND (B input)</b>	9.688	9.817
<b>NAND (Tall version, A input)</b>	9.3	9.61
<b>NAND (Tall version, B input)</b>	6.2	9.106
<b>XOR</b>	9.079	9.079
<b>PG4</b>	8.82975	6.717

## Section 7: Measurements in LTSPICE for Delays

In this section, we will discuss the results of the SPICE code shown in the previous section for both schematic and layout.

### Section 7.1: Schematic

The input and output waveforms generated by the Section 6 SPICE code for the 32-bit CLA schematic are shown below. Each figure shows 16 bits each.

Just to reiterate, the SPICE Code also performs the following numerical verification.

$$0xFFFFFFFF + 0x00000000 = 0xFFFFFFFF \text{ (Hexadecimal)}$$

$$-1 + 0 = -1 \text{ (signed decimal)}$$

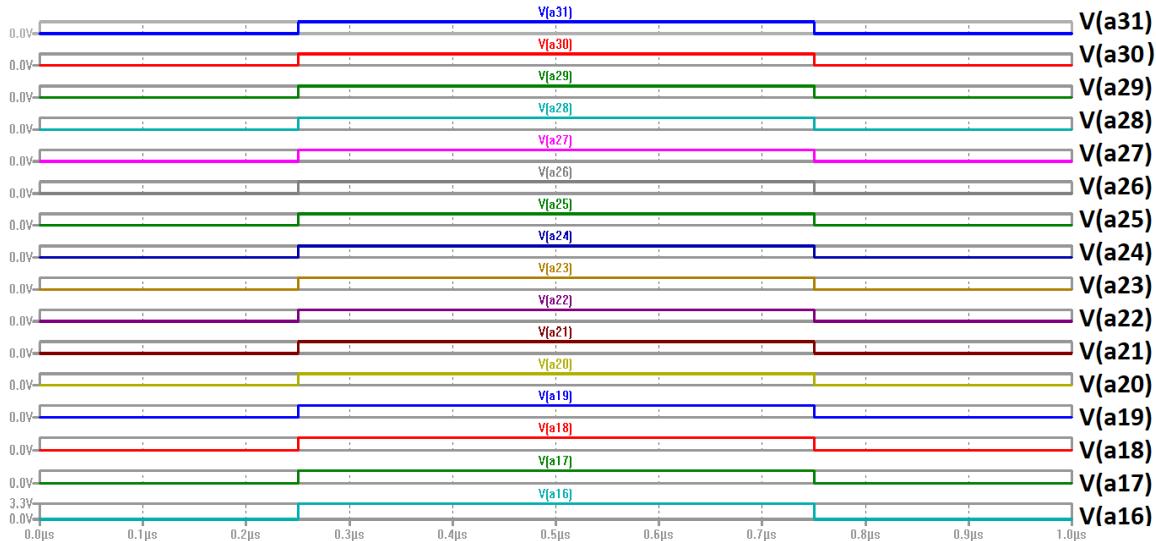


Figure 68: Input A[31:16]

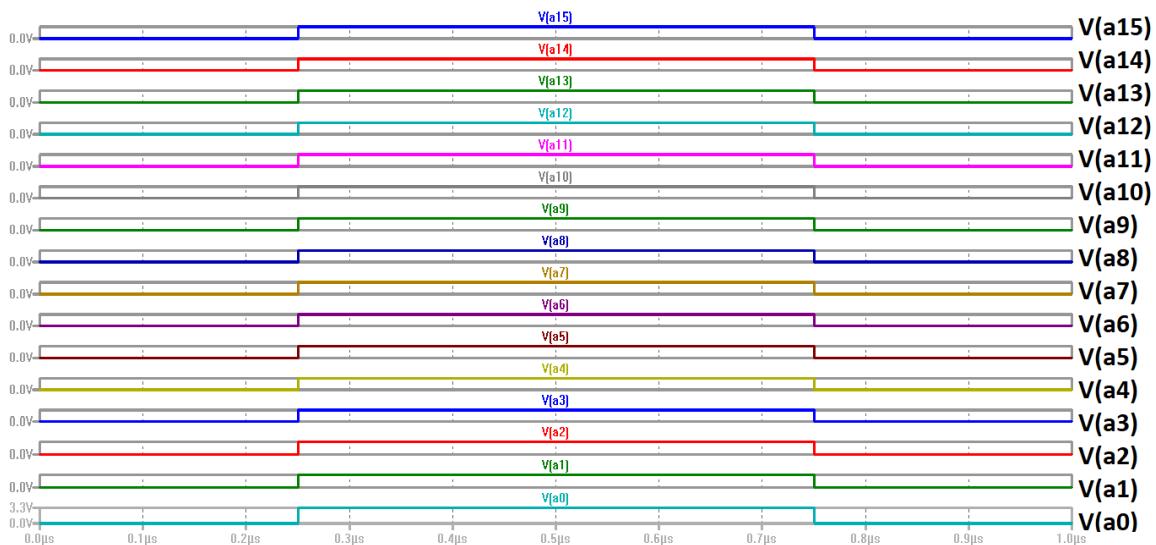


Figure 69: Input A[15:0]



Figure 70: Input B[31:16]

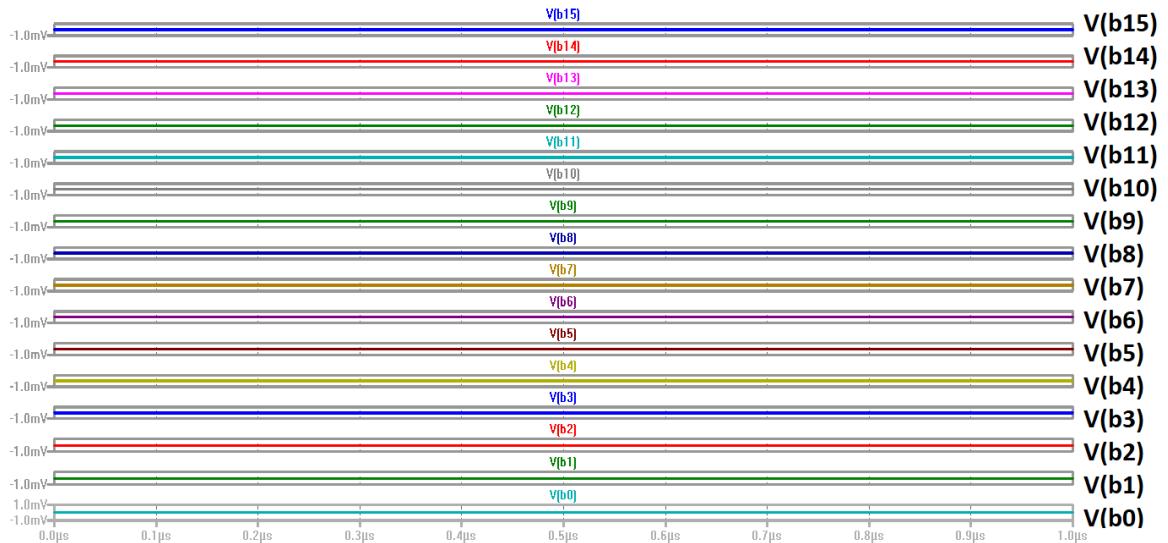


Figure 71: Input B[15:0]

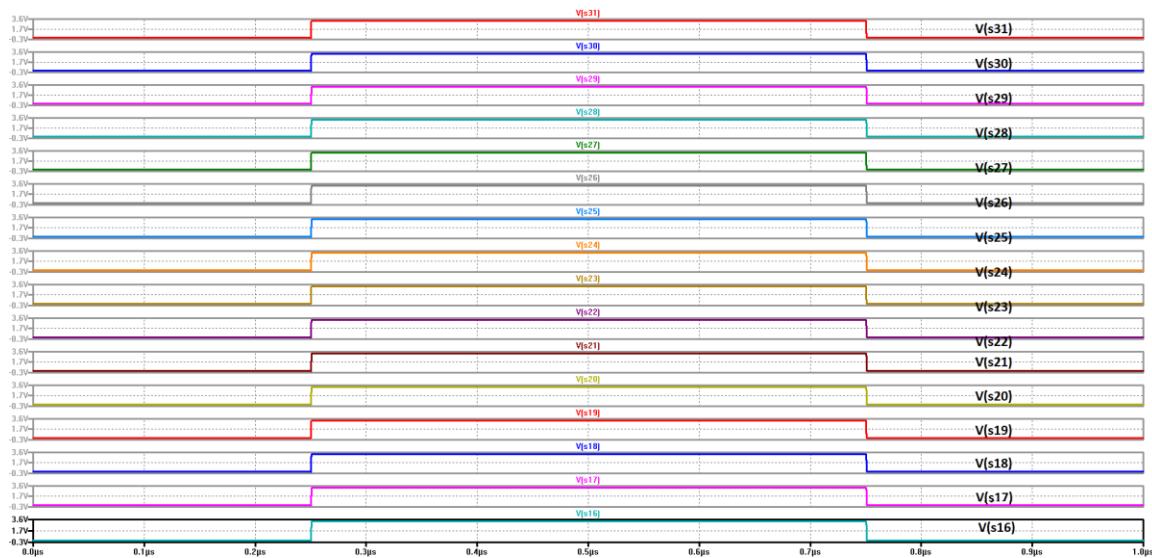


Figure 72: Schematic LTSPICE - Output S[31:16]

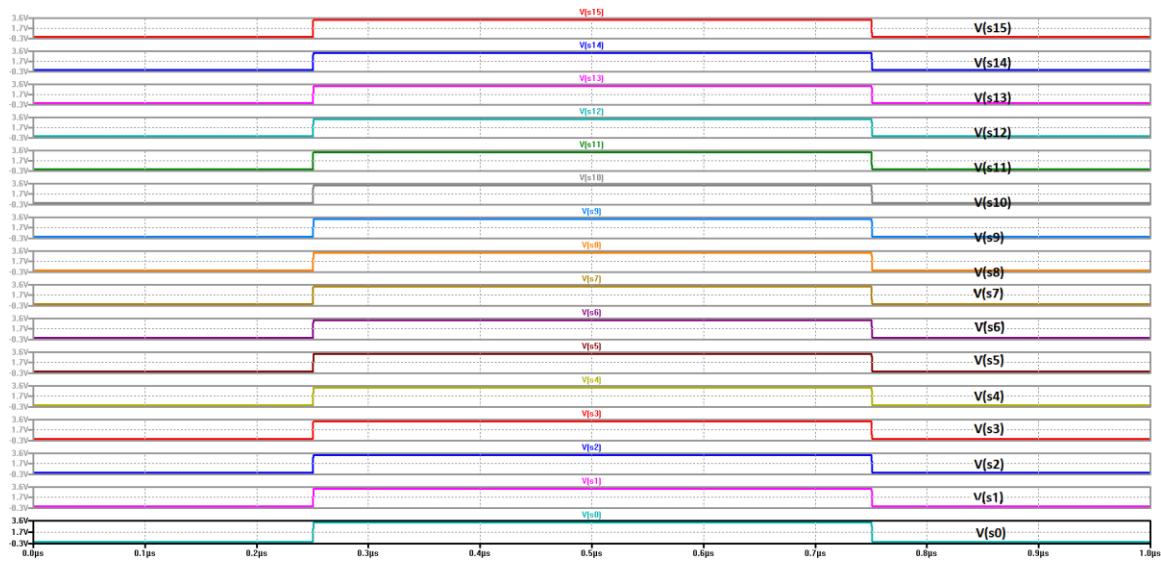


Figure 73: Schematic LTSPICE - Output S[15:0]

These are the results gathered by the .measure commands. Note that all values are in picoseconds.

**Table 5: Schematic - Rise Time, Fall Time, and Propagation Delay for All Bits**

Bit	Rise Time (ps)	Fall Time (ps)	$t_{ph}$ (ps)	$t_{phl}$ (ps)	$t_p$ (ps)
<b>0</b>	99.08	97.81	467.63	427.63	447.6255
<b>1</b>	103.41	101.43	481.84	438.37	460.1045
<b>2</b>	99.76	101.11	437.52	403.74	420.634
<b>3</b>	101.63	98.98	393.08	367.27	380.1725
<b>4</b>	98.07	100.29	467.68	426.55	447.116
<b>5</b>	103.42	101.45	481.84	438.38	460.1085
<b>6</b>	99.76	101.11	437.52	403.75	420.6385
<b>7</b>	101.63	98.98	393.08	367.27	380.177
<b>8</b>	98.07	100.29	467.68	426.51	447.0975
<b>9</b>	103.42	101.42	481.84	438.35	460.0955
<b>10</b>	99.76	101.12	437.52	403.73	420.6265
<b>11</b>	101.63	98.98	393.08	367.26	380.1695
<b>12</b>	98.07	100.21	467.68	426.48	447.081
<b>13</b>	103.42	101.41	481.84	438.32	460.081
<b>14</b>	99.76	101.11	437.52	403.69	420.605
<b>15</b>	101.63	98.97	393.08	367.24	380.1585
<b>16</b>	97.61	99.92	467.58	426.89	447.233
<b>17</b>	103.42	101.45	481.84	438.37	460.105
<b>18</b>	99.76	101.11	437.52	403.74	420.634
<b>19</b>	101.63	98.98	393.08	367.27	380.1725
<b>20</b>	98.08	100.29	467.69	426.55	447.1165
<b>21</b>	103.42	101.45	481.84	438.38	460.109
<b>22</b>	99.76	101.11	437.52	403.75	420.6385
<b>23</b>	101.63	98.98	393.08	367.27	380.177
<b>24</b>	98.07	100.29	467.68	426.51	447.0975
<b>25</b>	103.42	101.42	481.84	438.35	460.0955
<b>26</b>	99.76	101.12	437.52	403.73	420.6265
<b>27</b>	101.63	98.98	393.08	367.26	380.17
<b>28</b>	98.07	100.21	467.68	426.48	447.081
<b>29</b>	103.42	101.41	481.84	438.32	460.081
<b>30</b>	99.76	101.11	437.52	403.69	420.605
<b>31</b>	101.63	98.98	393.08	367.24	380.159

Below are the average values across all bits:

- Average Rise Time: 100.74 ps
- Average Fall Time: 100.36 ps
- Average Propagation Delay between Input and Output: 427.0185 ps

## Section 7.2: Layout

Below are the waveforms generated by the Section 6 SPICE code for the 32-bit CLA layout. Because the same input waveforms are used, only the output sum bits are shown for the layout. Even from the graphs, the output rise and fall times are noticeably less steep than the schematic waveforms.

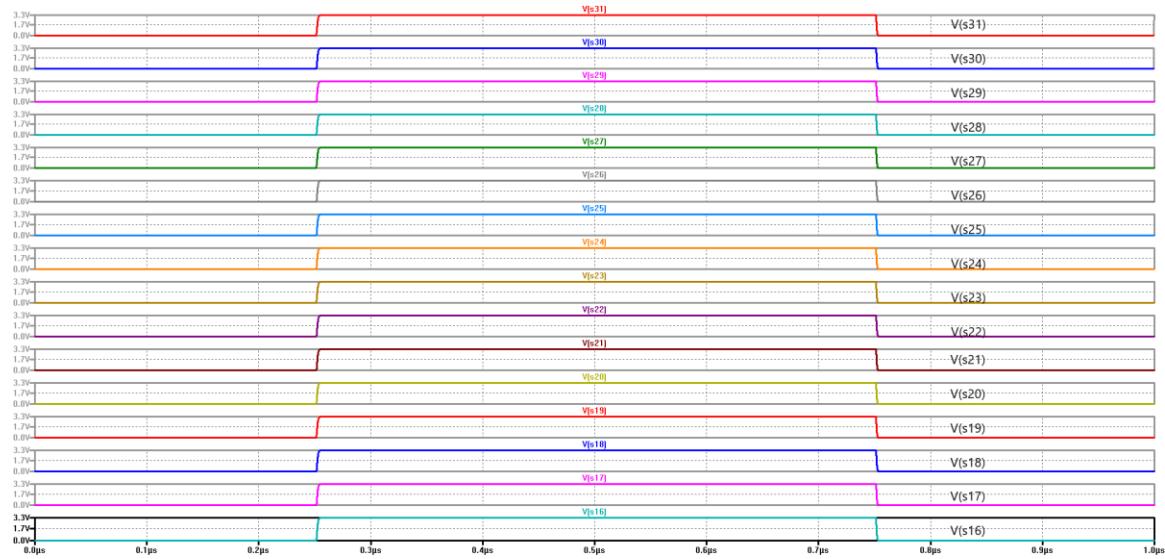


Figure 74: Layout LTSPICE - Output S[31:16]

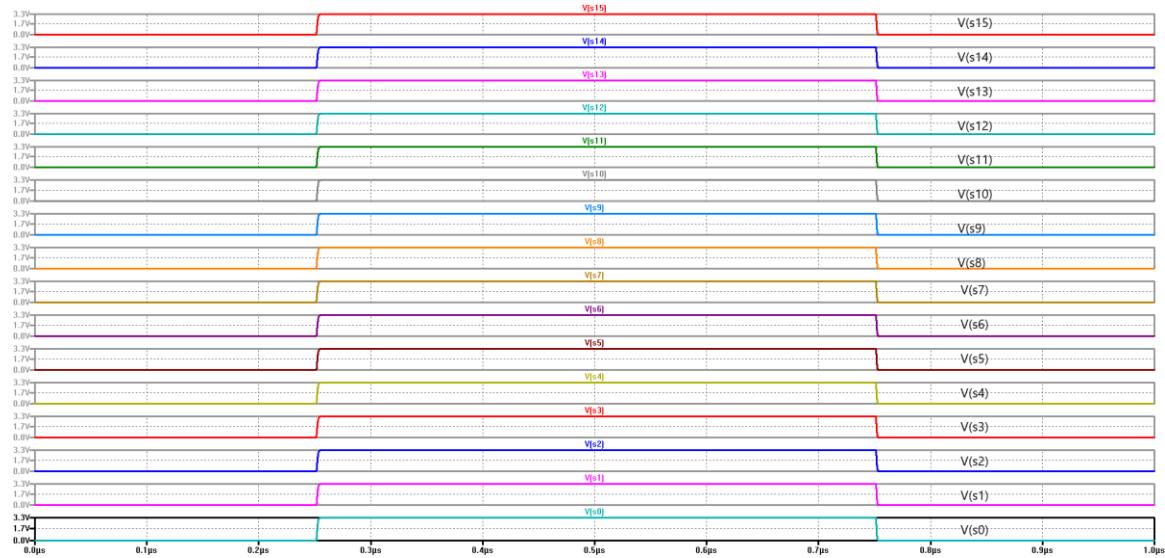


Figure 75: Layout LTSPICE - Output S[15:0]

Below are the metrics gathered by the .measure commands in the SPICE code. This time, all units are in nanoseconds.

**Table 6: Layout - Rise Time, Fall Time, and Propagation Delay for All Bits**

Bit	Rise Time (ns)	Fall Time (ns)	tplh (ns)	tphl (ns)	tp (ns)
0	1.6947	1.0267	2.1432	1.6739	1.908515
1	1.7027	1.0296	2.2131	1.7266	1.969865
2	1.7836	1.0709	2.2570	1.7569	2.00693
3	1.8039	1.0846	2.1466	1.6678	1.90721
4	1.7148	1.0365	2.1549	1.6780	1.916425
5	1.7102	1.0329	2.2162	1.7285	1.97233
6	1.7861	1.0728	2.2582	1.7576	2.007895
7	1.8048	1.0850	2.1470	1.6680	1.90749
8	1.7176	1.0382	2.1561	1.6788	1.91745
9	1.7131	1.0342	2.2174	1.7292	1.97327
10	1.7894	1.0753	2.2599	1.7585	2.00918
11	1.8086	1.0868	2.1486	1.6689	1.908745
12	1.7210	1.0401	2.1576	1.6797	1.918685
13	1.7411	1.0466	2.2293	1.7364	1.98281
14	1.7919	1.0772	2.2611	1.7592	2.01013
15	1.8096	1.0873	2.1490	1.6692	1.909095
16	1.7274	1.0442	2.1601	1.6819	1.92098
17	1.7027	1.0296	2.2131	1.7266	1.969865
18	1.7776	1.0663	2.2541	1.7552	2.00463
19	1.7970	1.0813	2.1437	1.6661	1.904915
20	1.7088	1.0330	2.1520	1.6763	1.914135
21	1.7033	1.0299	2.2134	1.7268	1.97009
22	1.7801	1.0682	2.2553	1.7559	2.0056
23	1.7979	1.0817	2.1441	1.6663	1.905195
24	1.7116	1.0346	2.1533	1.6771	1.91517
25	1.7063	1.0312	2.2146	1.7275	1.971025
26	1.7835	1.0708	2.2570	1.7568	2.006885
27	1.8017	1.0835	2.1457	1.6672	1.90645
28	1.7150	1.0366	2.1548	1.6780	1.916405
29	1.7346	1.0437	2.2263	1.7347	1.980475
30	1.7860	1.0727	2.2582	1.7575	2.007835
31	1.8028	1.0841	2.1461	1.6675	1.9068

Below are the average values across all bits.

- Average Rise Time: 1.7540 ns
- Average Fall Time: 1.0568 ns
- Average Propagation Delay: 1.951 ns

### Section 7.3: Comparison

As with the previous project, the layout timing metrics are far slower than the schematic timing metrics. This is to be expected because the layout is riddled with parasitic effects while the schematic timing metrics are idealized (without parasitic effects).

When comparing average values, the output layout rise time is 17.41 times slower than its schematic value. The output layout fall time is 10.53 times slower than its schematic counterpart. Finally, the average propagation delay is 4.57 times slower than its schematic counterpart.

## Section 8: Measurement of Power, Chip Area, and Transistor Count

In this section, we measure some more metrics that aren't related to the timing of the circuit. These include steady-state power dissipation, chip area, and total transistor count. The other measurements when it comes to timing and delay have already been measured and analyzed in Section 7.

### Section 8.1: Power Measurement

Like in previous projects, we measure the current at  $V_{DD}$  to determine the steady-state power dissipation of the layout. The  $I(V_{DD})$  graph below was generated by the same SPICE code that was used to measure timing metrics. That is why two peaks are present because the current peaks as the inputs are switching. However, we will be measuring the current at the middle portion of that line, between the two peaks.

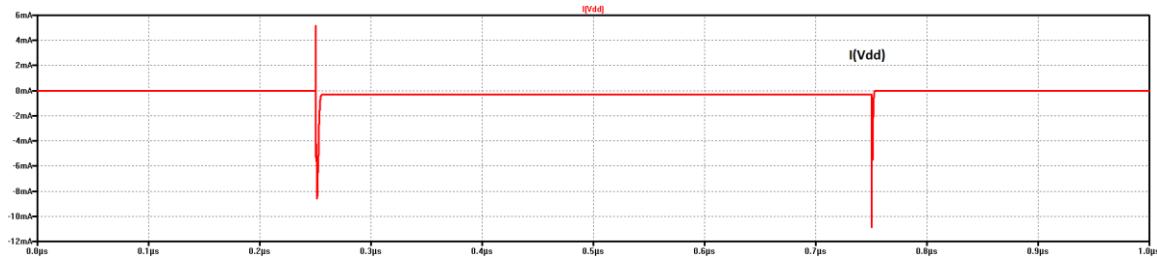


Figure 76:  $I(V_{DD})$  - Current at  $V_{DD}$  (Layout)

The steady state current was measured as  $-286.828\mu A$ . As with previous projects, we will use the absolute value of this current, as there is no such thing as negative power dissipation.

$$P = V_{DD} I_{DD}$$

$$P = 3.3 V * 286.828 \mu A$$

$$P = 946.5324 \mu W$$

### Section 8.2: Total Chip Area

Here, we use Electric VLSI's measuring tool to measure the  $dX$  and  $dY$  values of the 32-bit CLA layout. We then use those values to calculate the total chip area.

$$\begin{aligned} dX &= 1400\lambda; dY = 2886\lambda \\ A &= dX * dY \\ A &= 1400\lambda * 2886\lambda \\ A &= 4,040,400\lambda^2 \\ A &= 4,040,400 * (.175 \mu m)^2 \\ A &= 123,737.25 \mu m^2 \end{aligned}$$

### Section 8.3: Total Transistor Count

Like in previous projects, the basic building blocks that made this circuit possible are the low-level CMOS gates that were designed in previous projects. Specifically, the gates used were inverters, two-input NAND gates, and XOR gates. Below are the transistor counts for those gates.

$$T_{inv} = 2$$

$$T_{nand} = 4$$

$$T_{xor} = 12$$

The C1 module uses two NAND gates and one inverter. The C2 module uses four NAND gates and two inverters. The C3 module uses six NAND gates and three inverters. The PG4 module uses one inverter, as well as eight transistors to realize the four input AND function. Their transistor counts are shown below.

$$T_{C1} = 2(4) + 2 = 10$$

$$T_{C2} = 4(4) + 2(2) = 20$$

$$T_{C3} = 6(4) + 3(2) = 30$$

$$T_{PG4} = 8 + 2 = 10$$

The modified full adder itself uses two XOR gates, a NAND gate, and an inverter. Its transistor count is shown below.

$$T_{FA} = 2(12) + 4 + 2 = 30$$

With all these components lined up, we then calculate the transistor count of the four-bit CLA module, which contains 4 full adders, a C1 module, a C2 module, two C3 modules, and a PG4 module. Below is its transistor count.

$$T_{CLA-4} = 4(30) + 10 + 20 + 2(30) + 10$$

$$T_{CLA-4} = 220$$

The 16-bit CLA module uses four 4-bit CLAs, a C1 module, a C2 module, two C3 modules, and a PG4 module. Below is its transistor count.

$$T_{CLA-16} = 4(220) + 10 + 20 + 2(30) + 10$$

$$T_{CLA-16} = 980$$

The 32-bit CLA module uses two 16-bit CLAs and a C1 module. Below is the final transistor count for the entire circuit.

$$T_{CLA-32} = 2(980) + 10$$

$$T_{CLA-32} = 1970$$

### Section 9: Conclusion

In conclusion, by increasing the complexity of the adder circuit, we were able to decrease the time it took to perform binary addition. This is the core of the Carry Look-Ahead Adder, and it is its primary advantage against its simpler counterpart, the Ripple-Carry Adder. Here, we see the Space-Time Tradeoff at work. We increase the size and complexity of our circuit in order that it may perform binary addition faster than a 32-bit Ripple Carry Adder.

We also observe just how much parasitic resistors and capacitors can affect the performance of the circuit. As seen in Section 7, the output rise and fall times for the schematic were both approximately 0.1 ns for all 32 outputs. That number goes up to 1 ns in the layout! For the propagation delays, the number goes from 0.4 ns to 2 ns, almost 5 times slower!

### References

- [1] A. Singhal, "Getting the best of both worlds: Space-time trade-offs in algorithms.," *Hackernoon*. [Online]. Available: <https://hackernoon.com/getting-the-best-of-both-worlds-space-time-trade-offs-in-algorithms-b62116aaf3ef>. [Accessed: 08-Dec-2019].
- [2] S. Das, "Carry Look-Ahead Adder," GeeksforGeeks, 25-Nov-2019. [Online]. Available: <https://www.geeksforgeeks.org/carry-look-ahead-adder/>. [Accessed: 08-Dec-2019].
- [3] J. V. der Spiegel, "Carry Look-Ahead Adder," Carry Look Ahead Adder. [Online]. Available:  
<https://web.archive.org/web/20110925185302/http://www.seas.upenn.edu/~ese171/lab/CarryLookAhead/CarryLookAheadF01.html>. [Accessed: 15-Dec-2019].