VLSI Course Project: Design and Simulation of Four-Bit Adder Subtractor

EGC 446 VLSI Design Lab

Group Members	Department	ID Number
George Dagis	CE	N03665861

May 16th, 2018

Instructor: Damu Radhakrishnan

Abstract

For the final project, a four-bit adder subtractor was created by using Electric VLSI software to construct various schematics, layouts and icons. It is known that a four bit adder-subtractor has 9 inputs: A3-A0, B3-B0 and Sub. The four bit adder-subtractor also has 6 outputs: Sum3-Sum0, Carry Out and a Sign bit. Each combination of inputs were tested in order to verify proper functionality of the adder subtractor-circuit.

Table of Contents

1. Introduction ……..……..... 1

2. Design Procedure……...……...……...…. 2

2.1 One-Bit Full Adder ……...…..……..……..... 2

2.12 Transistor Schematics ..……....……..... 4

2.13 Simulations of CMOS ……..... 7

2.14 Layout Design ……....……....……..... 8

2.15 Layout Simulation …..……..... 13

2.16 Full Adder Icon ……....……....……..……..... 14

2.2 Four-Bit Adder …….……..... 14

2.21 Layout Design ……...……....……..... 16

2.22 Fan-Out-Four ……..……..... 19

2.23 Layout Simulation ……....……....……..……..... 20

2.24 Four-Bit Adder Icon ….....……....……..……..... 23

2.3 Four Bit Adder Subtractor ……...……... 23

2.31 Transistor Schematics ..……....……..... 24

2.32 Layout Design …..……....……....……..……..... 25

2.33 Padframe Layout ...……....……....……..……..... 27

3. Verification …...……..... 31

4. Measurements & Analysis ……...……..……..... 36

5. Conclusion ...……...……..... 42

6. Bibliography ...……..……..... 44

1. Introduction

The purpose of this project was to build and simulate a four bit adder-subtractor utilizing Electric VLSI tools. Multiple CMOS schematics, layouts and icons were created and their functionality was compared with ideal versions of themselves. Schematics, layouts and icons were made for various logic gates, a full adder, a four bit adder, and a four bit adder-subtractor. These were all simulated and the logic was analyzed for verification purposes. The cells were simulated using LTSpice simulation. Throughout the project, the transistor models were created using C5 process - .5µm CMOS technology. The smallest NMOS transistor size 10/2 was used.

It is important to recall how the four bit adder-subtractor circuit works before attempting to build one. The four bit adder-subtractor circuit adds 2 four-bit numbers. These inputs are denoted as A3-A0 and B3-B0. A user may also specify whether they want the numbers to be added or subtracted. The bit to select addition or subtraction is denoted as "Sub." The four bit adder-subtractor also has 6 outputs. Four of these outputs are sum bits, S3-S0. The fifth output bit is a carry-out bit (or 'Cout'). The adder-subtractor also has a 'sign' bit denoting if the output is positive or negative. The four bit adder-subtractor can be created through the use of four one-bit adders, which simply takes one A, one B, and one Cin input. The carry output of a one bit adder is fed to the 'C' input or 'Cin' of the next [1]. This can be seen in the "One-Bit Full Adder" section. The four bit adder circuit is often used in cascade of other adders, allowing for addition of 2^n binary numbers. 8-bit adders, 16-bit adders, and so on can be created through cascading of these adders. A four-bit adder-subtractor diagram can be seen in Figure 1.

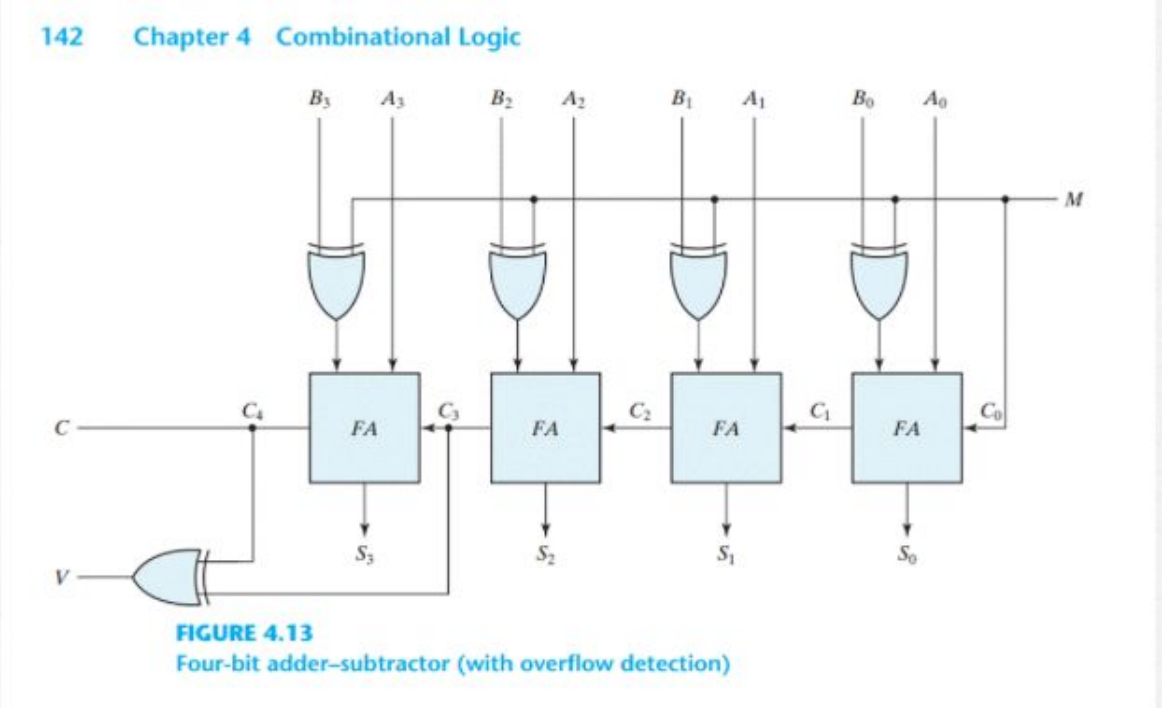

Figure 1: Four Bit Adder-Subtractor Circuit from 4 Full Adders

2. Design Procedure

2.1 One-Bit Full Adder

The boolean logic for the full adder was found using the truth table in Figure 2. This truth table shows inputs A, B and C for the full adder, as well as the corresponding outputs “sum” and “cout”.

A	B	C		Sum	Cout
0	0	0		0	0
0	0	1		1	0
0	1	0		1	0
0	1	1		0	1
1	0	0		1	0
1	0	1		0	1
1	1	0		0	1
1	1	1		1	1

Figure 2: Full Adder Truth Table

Using this truth table, it was easy to derive the KMAPs for the full adder outputs. The KMAP for the carry out of the full adder can be seen in Figure 3, and the KMAP for sum of the full adder can be seen in Figure 4. These KMAPs were created using an online website for clearer viewing [2].

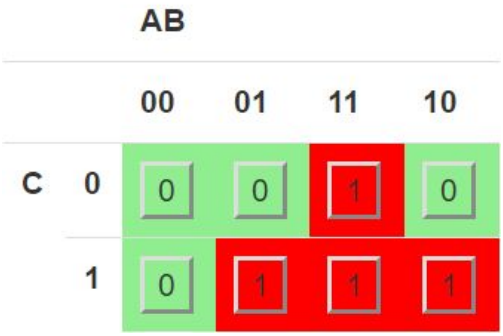

Figure 3: Full Adder "Carry Output" KMAP

Figure 4: Full Adder "Sum" KMAP

The boolean expression of the full adder sum is known to be A⊕B⊕C. The boolean expression for the carry output of the full adder was then found using three separate groupings The three groupings each contain the ABC cell as well as a different adjacent 1 cell. The logic expression for the carry output was found to be AB+AC+BC.

Euler paths were analyzed for each network and allowed for a resulting stick diagram to be created for the full adder. Dark blue was used for metal 1, light blue was used for metal 2, red was used for polysilicon, orange was used for PMOS and green was used for NMOS. The inputs A, B, C (Cin) as well as the outputs carry out (Co) and Sum (S) are also labeled. The resulting stick diagram and graph for the full adder circuit can be seen in Figure 5.

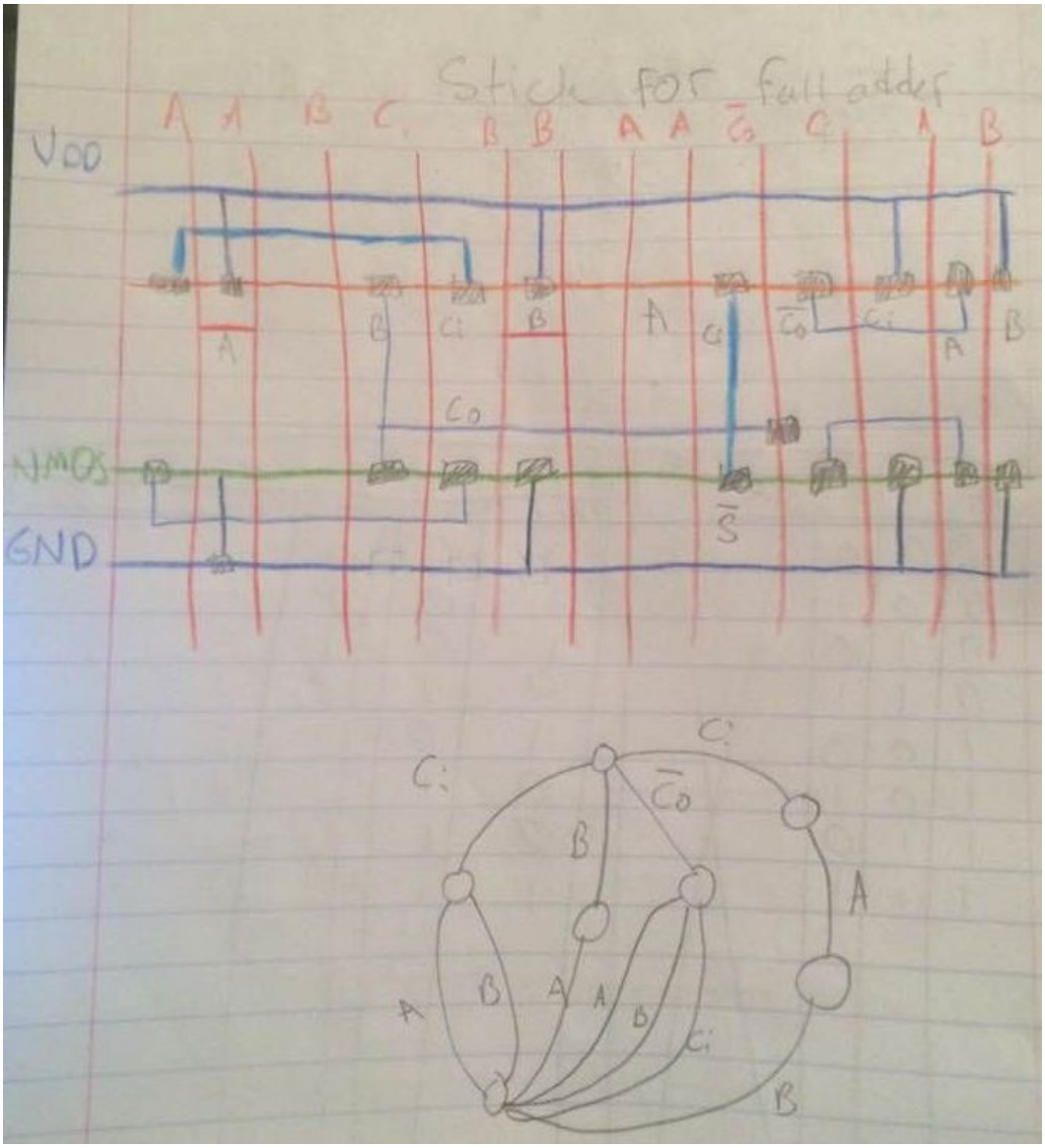

Figure 5: Stick Diagram and Graph for Full Adder

2.12 Transistor Schematic for One Bit Full Adder

The CMOS carry out schematic can be seen in Figure 6. The transistors are labeled accordingly, along with the transistor widths and lengths. Inputs A, B & Cin (C), as well as the output, Cout, are labeled. The three inputs all go through respective inverters in order to get the compliment signal, and are eventually connected to the gates of some transistors.

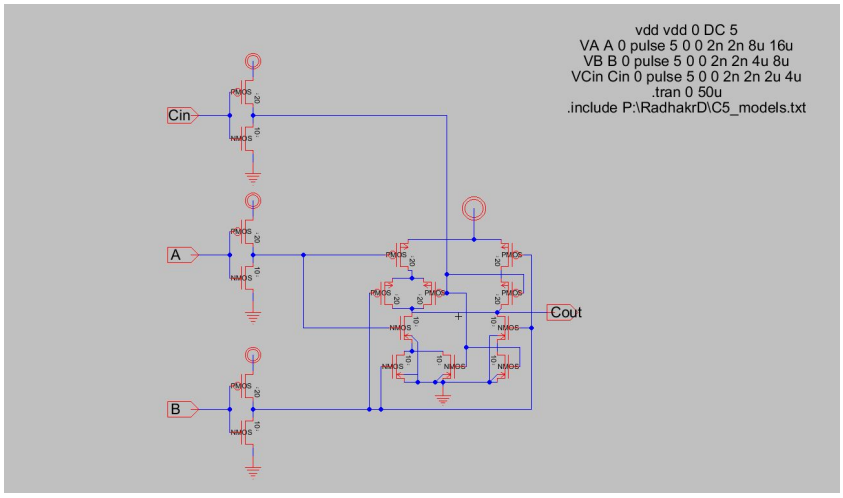

Figure 6: Full Adder Cout CMOS Schematic

The CMOS sum schematic can be seen in Figure 7. The transistors are labeled accordingly, along with the transistor widths and lengths. Inputs A, B & Cin, as well as the output, sum, are labeled. The three inputs all go through respective inverters in order to get the compliment signal, and are eventually connected to the gates of some transistors.

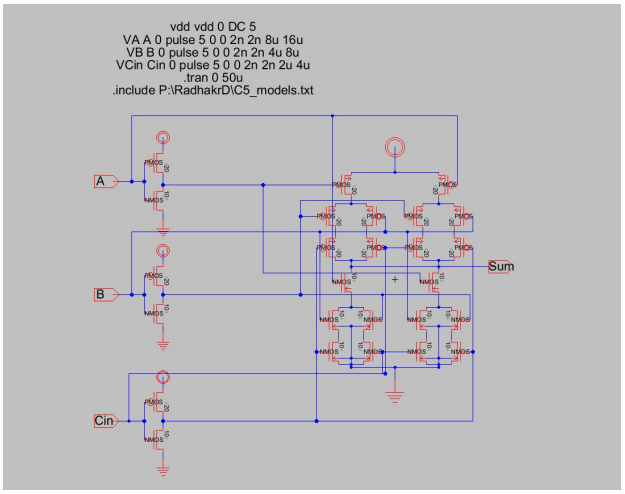

Figure 7: Full Adder Sum CMOS Schematic

After each of the two CMOS transistor schematics were designed and simulated using the proper spice code shown. The two were combined into one CMOS full adder schematic with

both Cout and Sum outputs. The complete full adder CMOS schematic can be seen in Figure 8. The transistors are labeled accordingly, along with the transistor widths and lengths. Inputs A, B & Cin, as well as the output, cout and sum, are labeled. The three inputs all go through respective inverters in order to get the compliment signal, and are eventually connected to the gates of some transistors as before. Similarly, spice code was implemented and can be seen above the transistor networks.

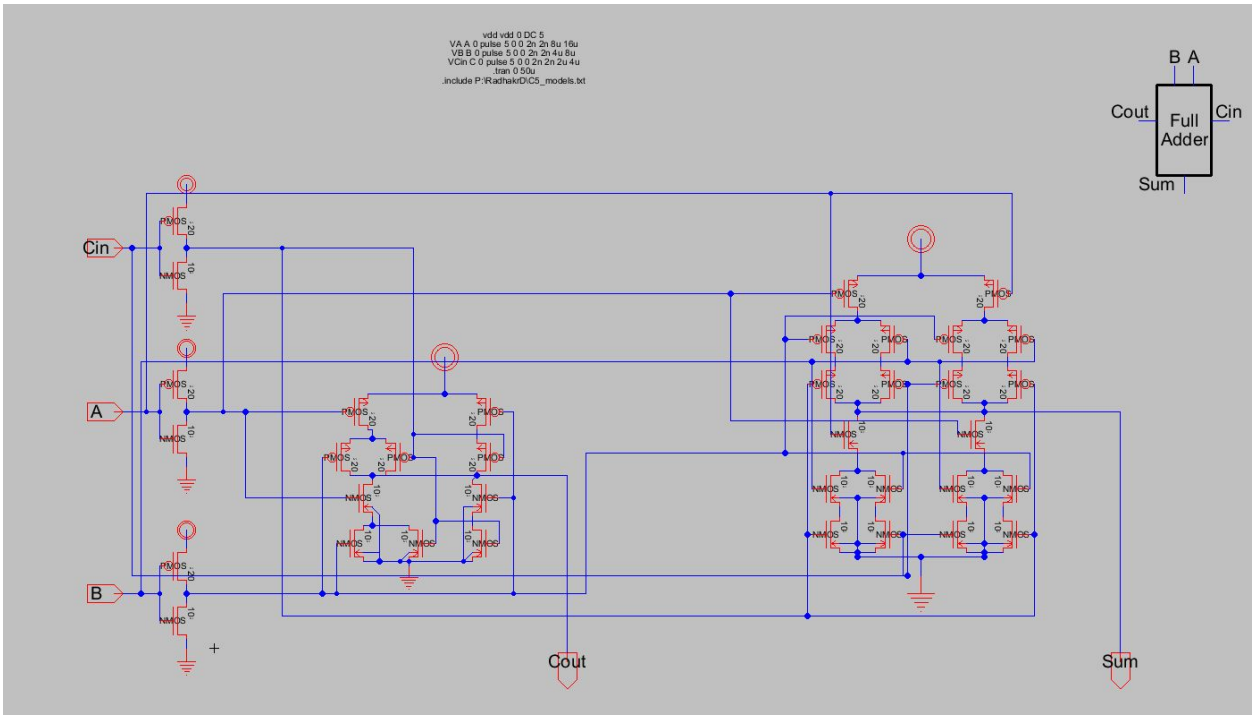

Figure 8: Completed Full Adder CMOS Schematic

2.13 Simulations of Input / Output Waveforms for CMOS Full Adder

After the final full adder schematic was created it was tested for verification purposes using LTSpice software. The same spice code was used for the three schematics as the inputs remained A, B, and Cin. The spice code sends high signals to each of the inputs at different times such that the output can be monitored for every possible binary input 000 to 111.

It is known that full adder adds three binary numbers and produces the output by using a sum and cout output. The sum output is a '1' when 001, 010, 100 or 111 are applied as inputs, and otherwise a '0'. That is, whenever only 1 or 3 input bits are 'high', the output for sum is also 'high'. The cout output is a '1' when 011, 101, 110 or 111 is applied as inputs, and otherwise a '0'. That is, whenever 2 or more of the 3 input bits are 'high', the output for cout is 'high'.

Output voltages were analyzed in the full adder as the input voltages were set to go from 0 (low) to high (1) at differently pulsed intervals. Transient simulations were performed of 50μs, meaning the transient simulation was performed from 0μs to 50μs. The PSPICE simulations for the cout and sum schematics were done individually before the completed full adder was created in order to avoid potential error. Figure 9 shows the output voltage of the completed full adder schematic. Signals are labeled for clarification purposes.

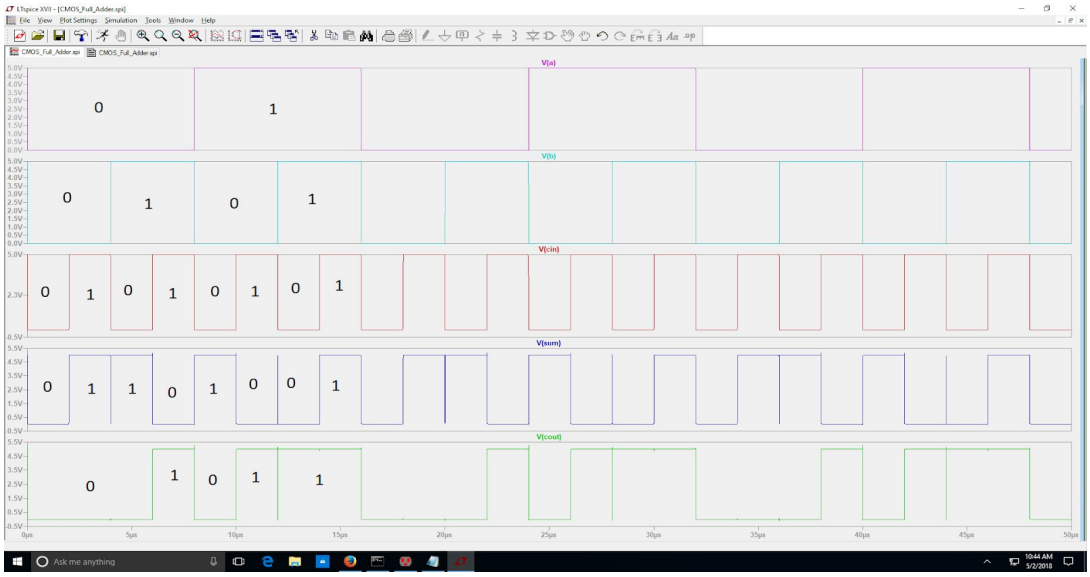

Figure 9: Full Adder Schematic Simulations

2.14 Layout Design of Full Adder

To make the full adder layout simple and compact, the layout was modeled using gate logic. Using 3 NAND gates and 2 XOR gates, the layout was able to be easily designed. The NAND schematic and icon can be seen in Figure 10. The NAND layout can be seen in Figure 11. Spice code is also shown.

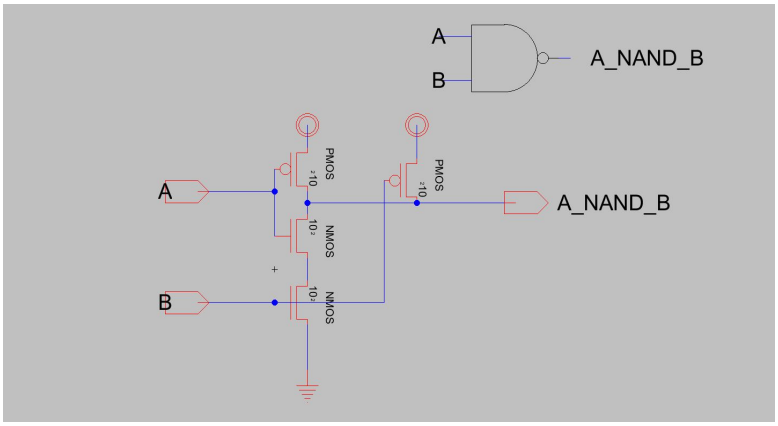

Figure 10: NAND CMOS Schematic and Icon

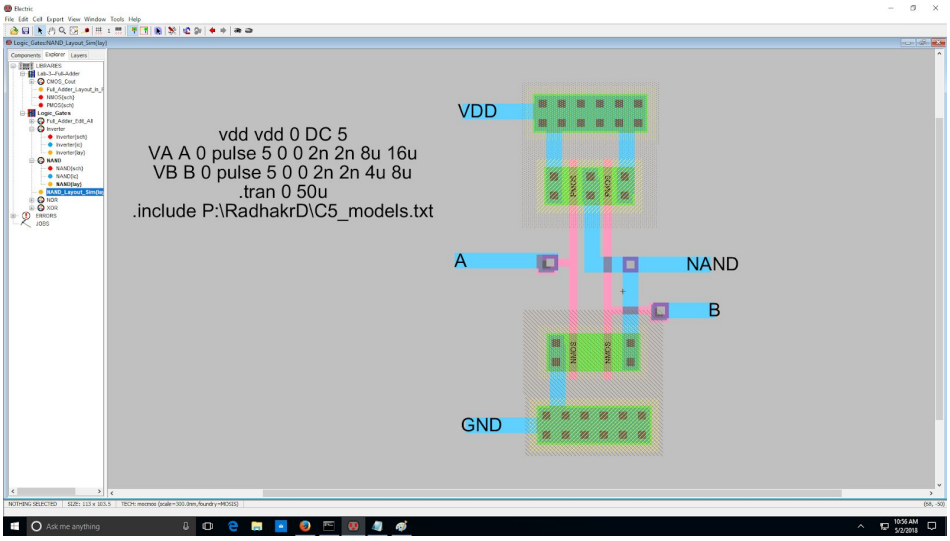

Figure 11: NAND Gate Layout

The XOR gate was then made using CMOS transistor design. The transistor schematic and icon can be seen in Figure 12. The XOR layout was then created and can be seen in Figure 13.

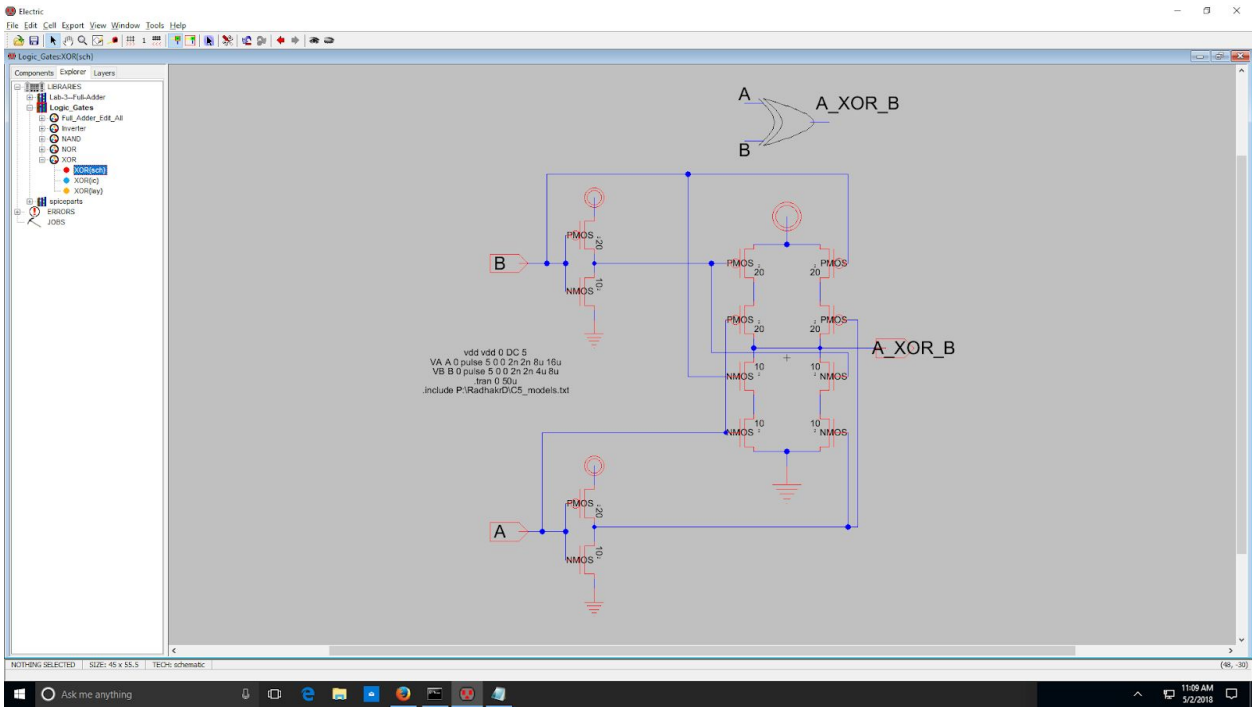

Figure 12: XOR CMOS Schematic and Icon

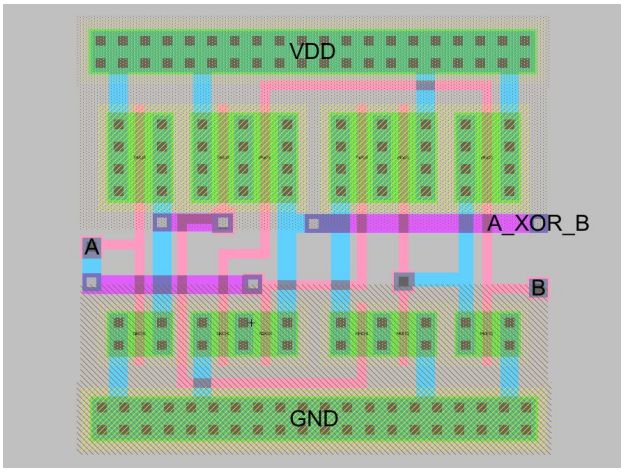

Figure 13: XOR Gate Layout

To get the logic expressions in terms of fewest gates, boolean algebra was used to manipulate the output logic expressions. NAND and XOR gates gave the best results. Because XOR gates were to be used in the final layout, only the cout logic needed to be manipulated, as the sum logic expression remained A⊕B⊕C. The carry output logic expression was manipulated using various techniques. The step by step manipulation as well as the truth table used for verification purposes are shown in Figure 14. This circuit could also be implemented with the original circuit, using 2 XOR gates, 3 AND gates and 2 OR gates.

Cout			**A⊕B⊕C**	**((AB)'(C(A⊕B))')'**
AB+AC+BC	Original		Actual Sum	Actual Cout
(AB)+(C(A+B))	Regroup		0	0
((AB)'(C(A+B))')'	Demorgans		1	0
((AB)'(C(A⊕B))')'			1	0
			0	1
			1	0
			0	1
			0	1
			1	1
A	B	C	Or	XOR
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	0

Figure 14: Derivation of Gate Logic

The table in Figure 14 with columns ‘Or’ and ‘XOR’ were meant to test a portion of the logic in the ‘regroup’ step (C(A+B)) with that of the final step (C(A⊕B))’ to confirm the logic was equal. For 7 out of the 8 inputs, the logic was the same, although when the input was 111 the logic was not equivalent. The input 111 was then substituted and checked in both the original and assumed expression, giving an equivalent answer. This equivalency is because the first term in the final expression, (AB)’ is NAND’ed with the second term.

After the logic expression was verified to give the same output logic, the circuit was made. The circuit along with the icon and proper spice code can be seen in Figure 15.

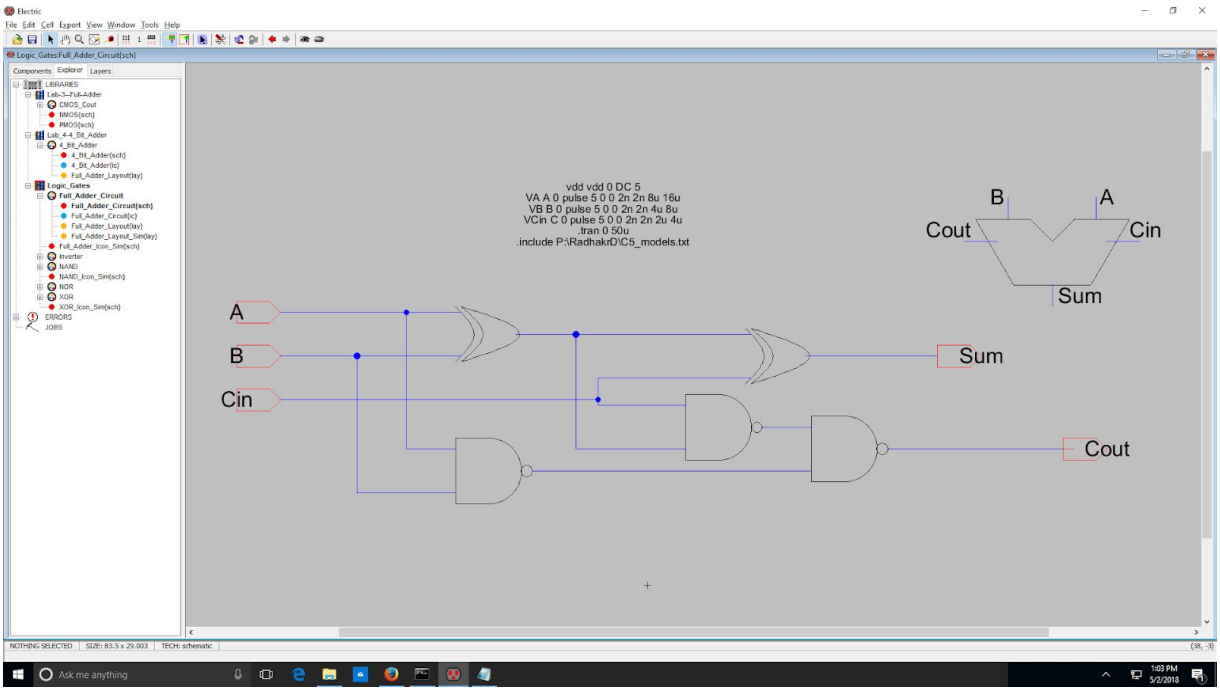

Figure 15: Circuit for Full Adder using NAND and XOR gates

After the circuit was simulated using previously mentioned techniques and proved to output the expected outputs, the layout was then ready to be created. When compiling the final full adder layout, first the gates were laid down. Next the VDD and GND bus were added, as

well as the inputs with metal 1. Then the final layout was made. This process can be seen in Figures 16, 17 and 18.

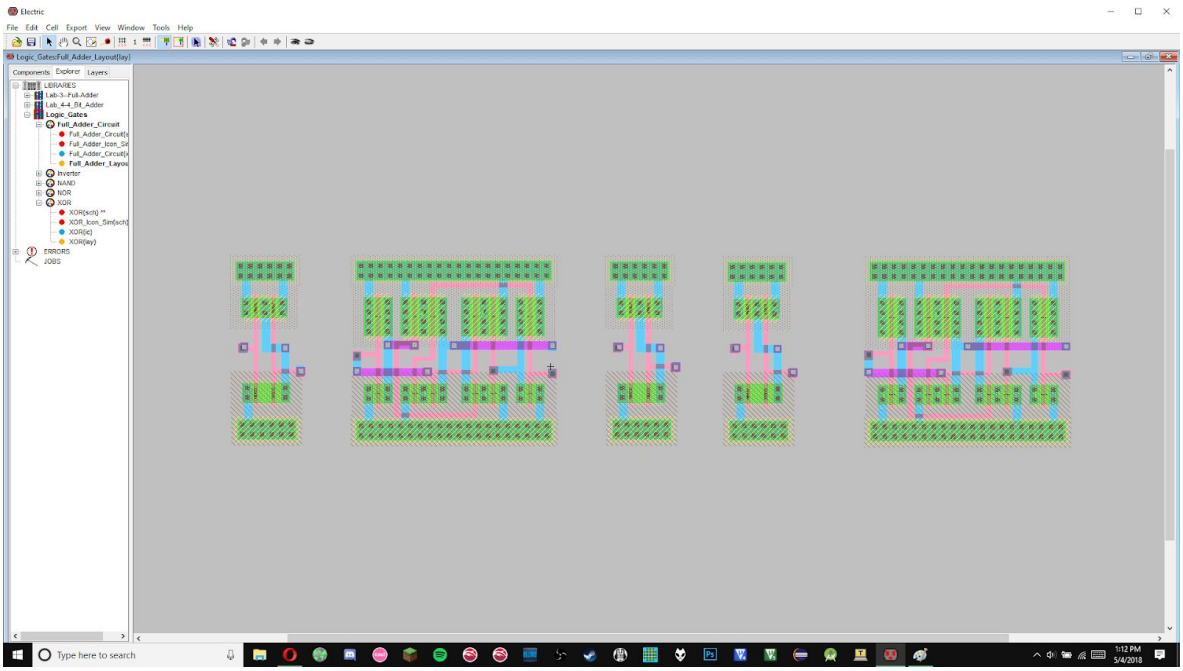

Figure 16: Full Adder Layout Gates

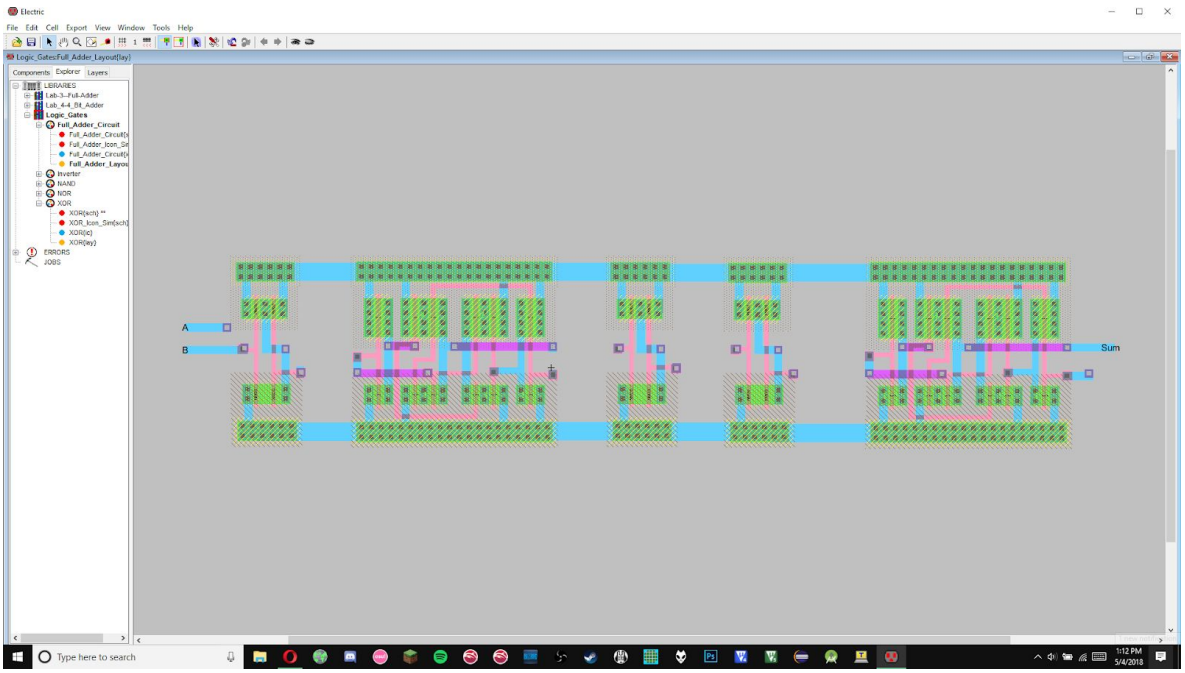

Figure 17: Full Adder Layout VDD, GND, Inputs Connected

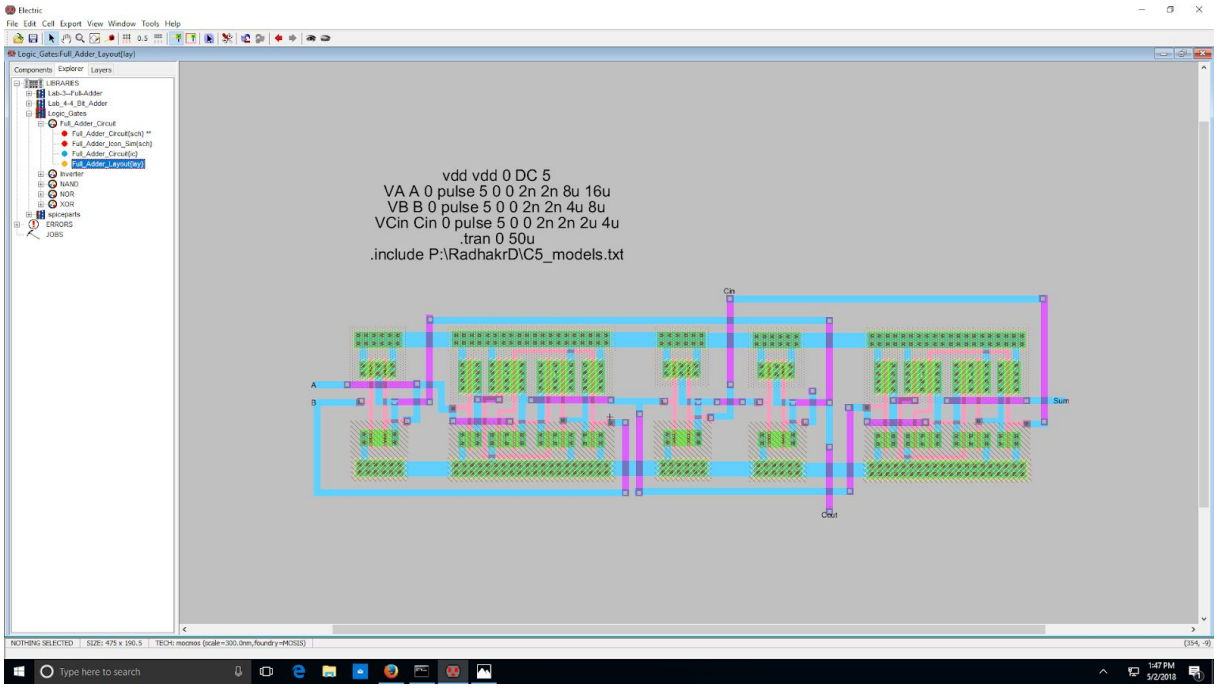

Figure 17: Completed Full Adder Layout

2.15 Simulated Input / Output Waveforms for Full Adder Layout

After the layout was completed, it was simulated using the code shown in Figure 17. The output waveform plot can be seen in Figure 18. The full adder layout performs as expected.

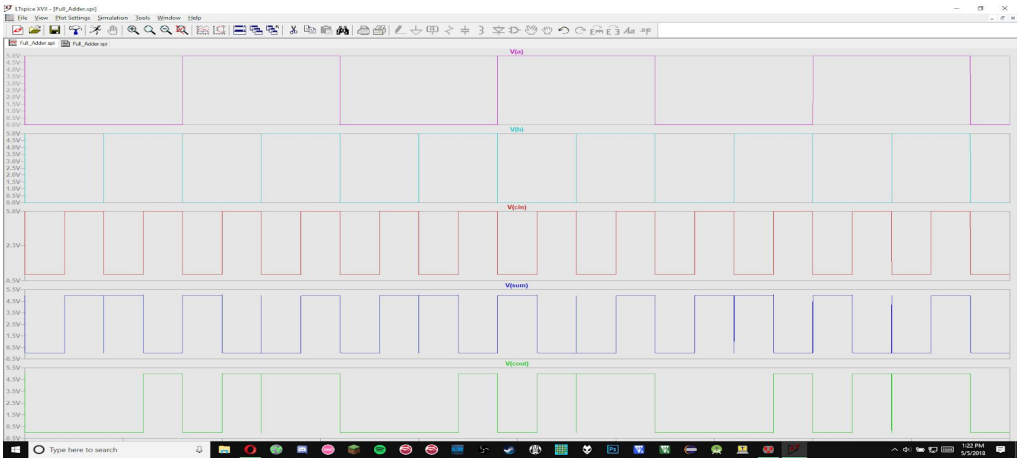

Figure 18: Full Adder Layout Simulation

2.16 Full Adder Icon

Because both the full adder circuit is a complex circuit and is used often, an icon was created to show a more simplistic view. An adder icon was made for the full adder. This icon can be seen in Figure 19. There is spice code included for simulation purposes.

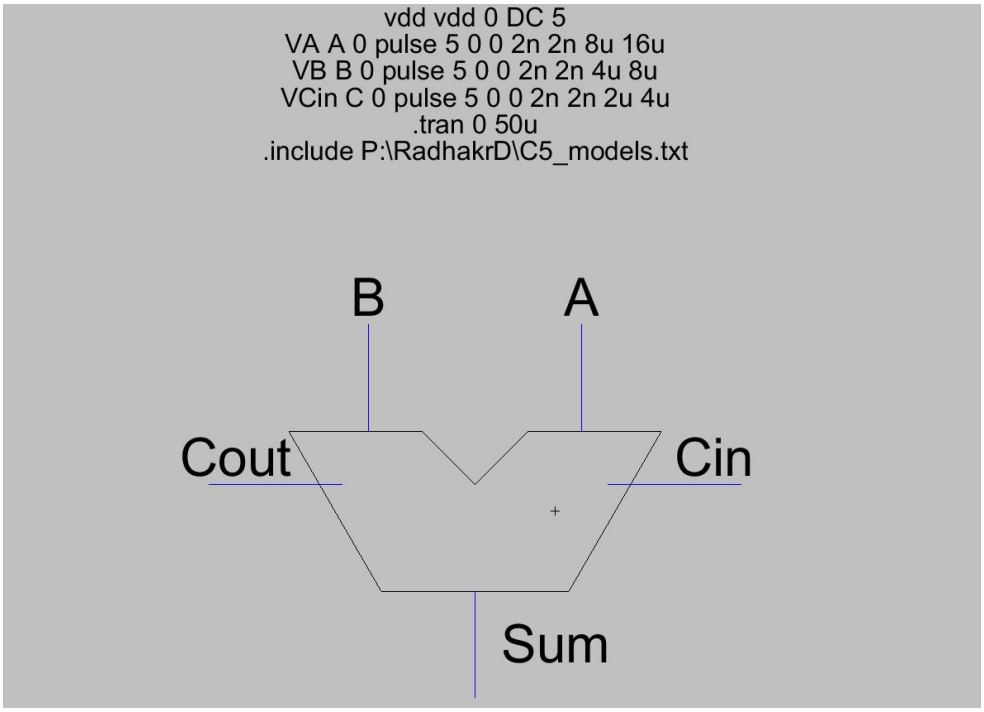

Figure 19: Full Adder Icon

2.2 Four-Bit Adder

To create a four-bit adder, the diagram shown in figure 17 must be understood. The one bit adder schematic was simply copied four times, and the cout of the first three one bit adder groups were connected to the cin of the next. The sums were then labeled in increasing order,

starting with ‘S0’ and going to S1’. The cout output of the fourth four bit adder is also labeled accordingly. The CMOS transistor schematic of the four bit adder can be seen in figure 20.

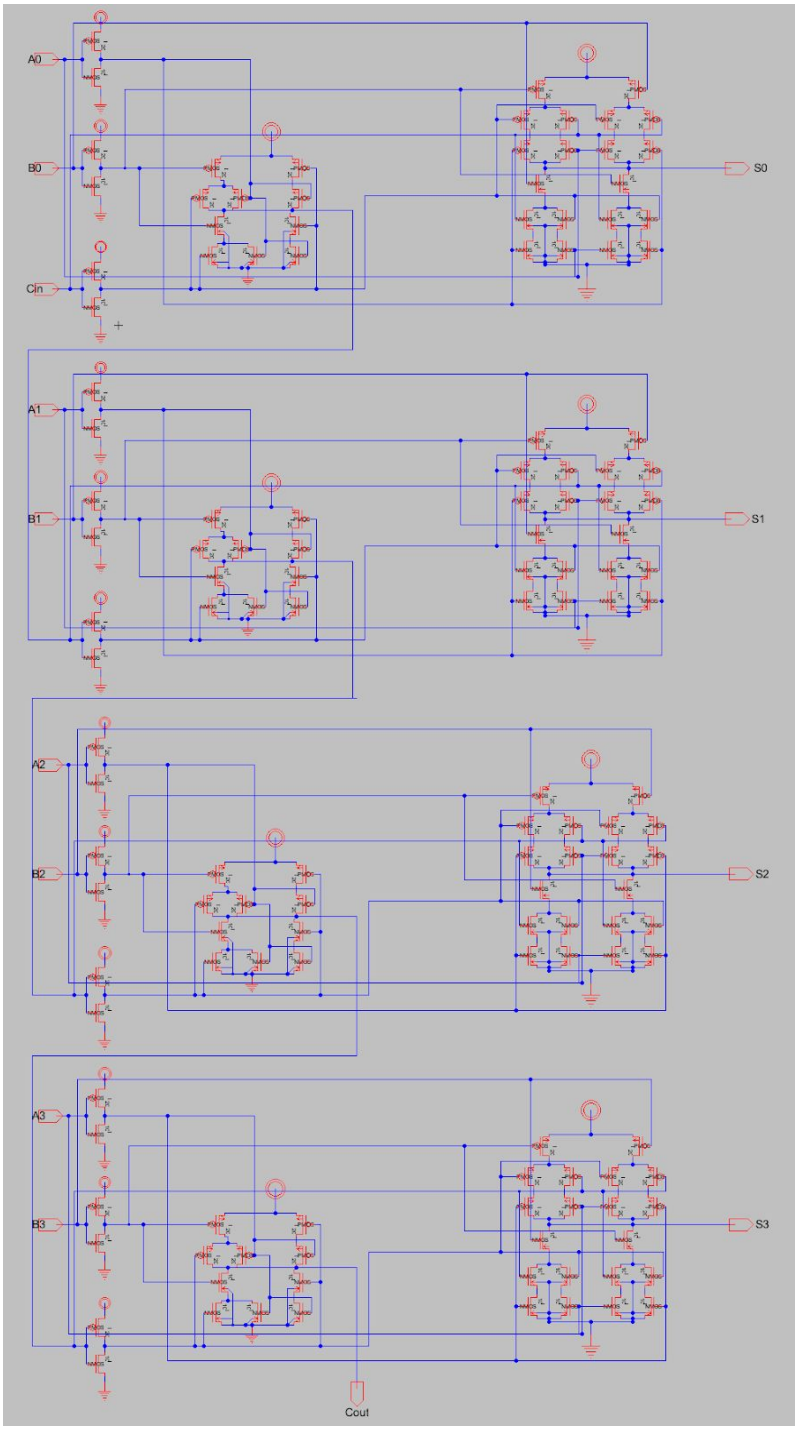

Figure 20: CMOS Transistor Schematic for Four Bit Adder

2.21 Layout Design of Four Bit Adder

To begin the layout of the four bit adder, the layout of the one bit adder was duplicated and added three more times, and then slight adjustments were made. In figure 21, the one bit adder layout is shown. The view is one level up from the lowest layer. This view is for clarity.

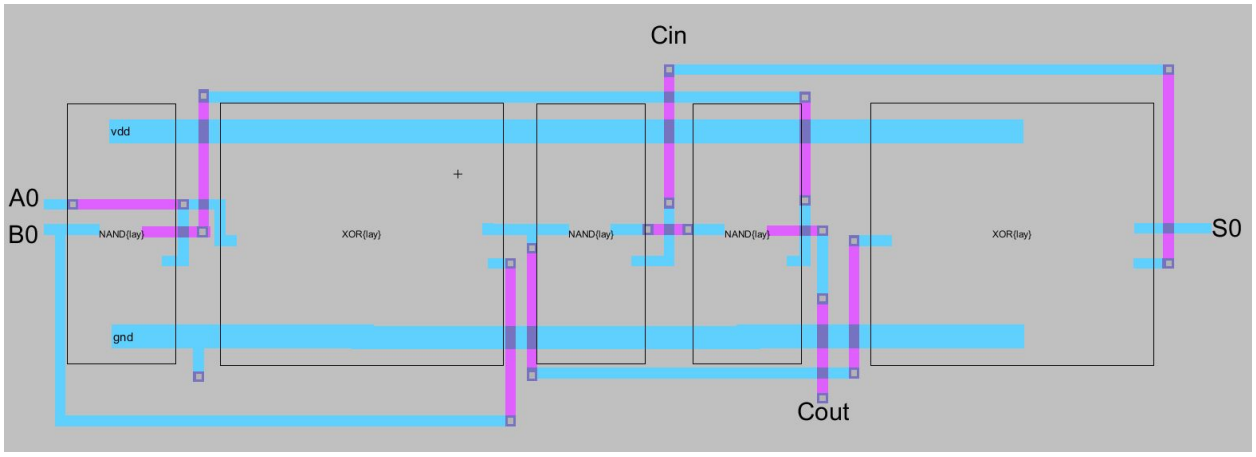

Figure 21: One Bit Adder Layout

After the one bit adder layout was placed in the new “four bit adder” layout cell, it was duplicated to create for one bit adder modules. They were placed evenly spaced apart and the sums and inputs were labeled appropriately, with the first input bits being A0 and B0, and ending with A3 and B3. After the exports were made for the inputs and sums, the cins and couts were connected appropriately, with the first input labeled ‘cin’ and last output being ‘cout’. The nodes where the these connections were made were labeled C0, C1 and C2. This was done for debugging purposes if simulations did not give expected waveforms. This can be seen in Figure 22.

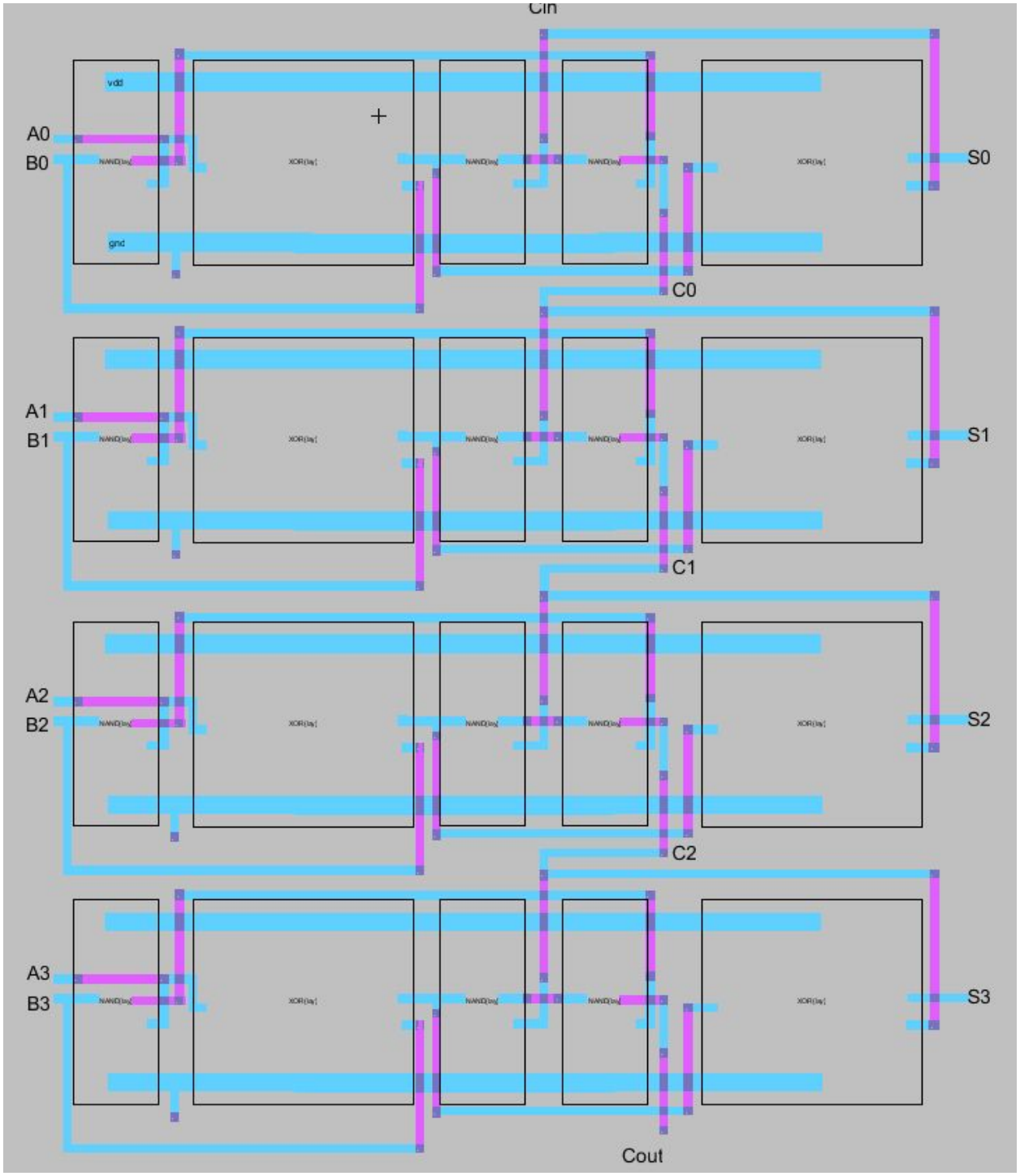

Figure 22: Four Bit Adder Layout with Carry's Labeled

Finally, the powers and grounds of each adder were connected to the powers and grounds of the other adders. The four bit adder was also rotated 90° to replicate the common depiction of four adders as shown in diagrams such as Figure 1. This new layout is in Figure 23.

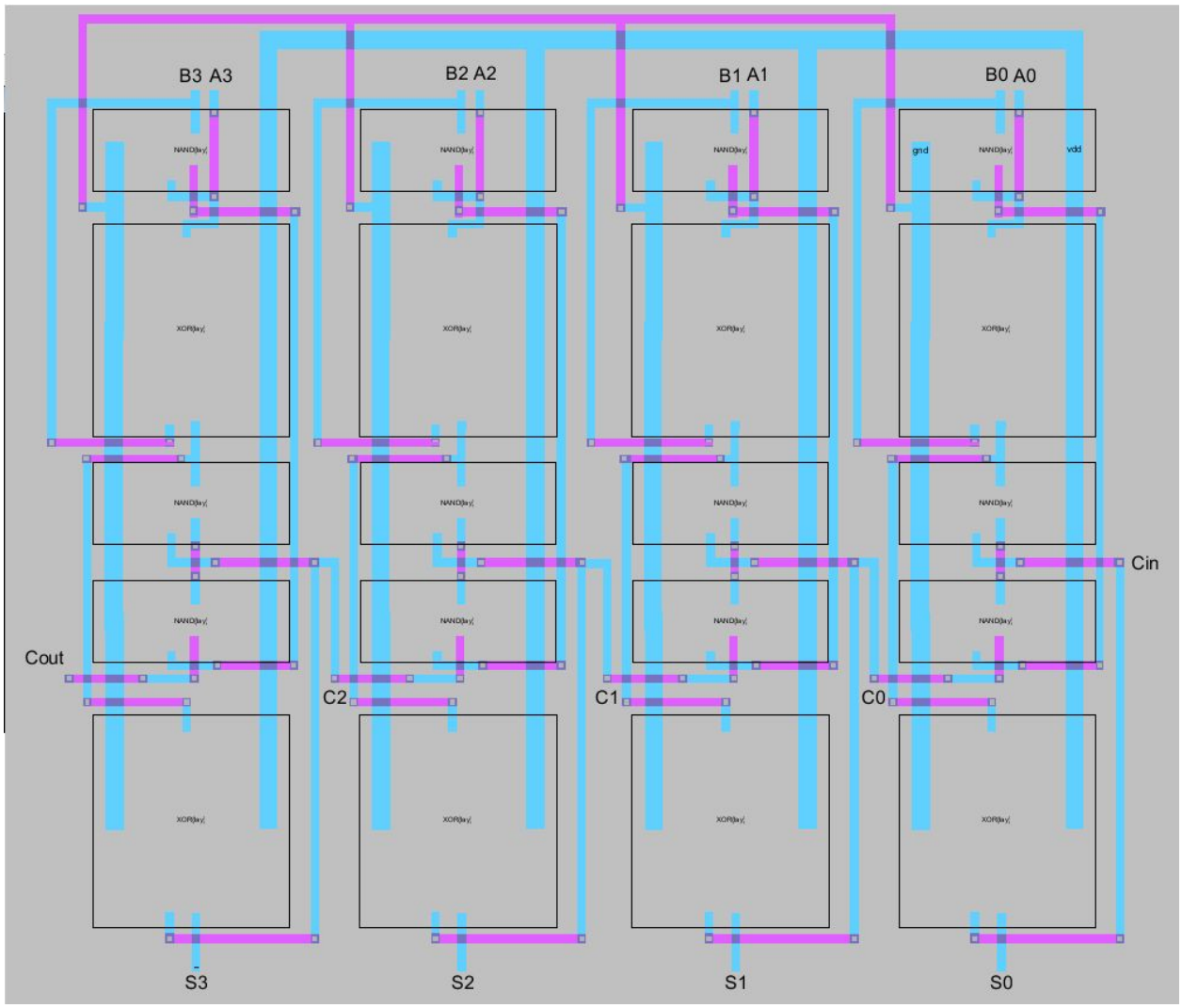

Figure 23: Finished Four Bit Adder Layout

2.22 Fan-Out-Four Load

In the industry, cells are often simulated using some load connected to the output. Because of this, a fan-out-of-four (FO4) load was created using four unit sized inverters. The load was created by connecting 4 parallel inverters. The FO4 load can be seen in figure 24.

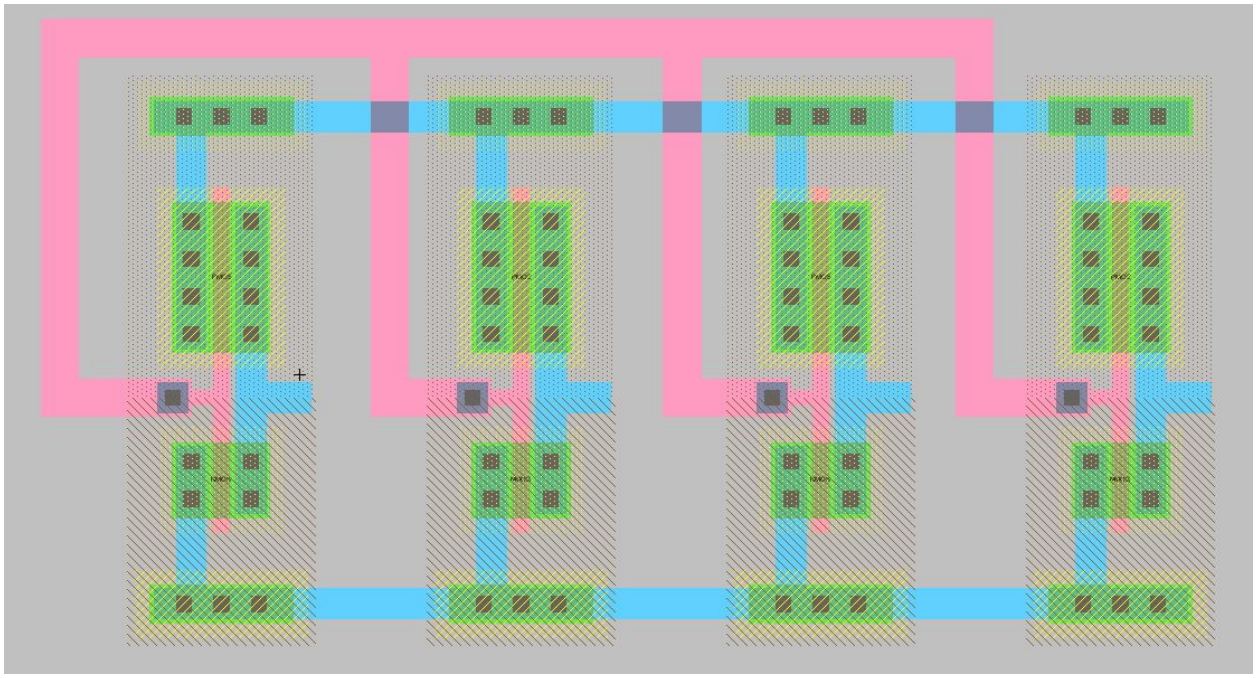

Figure 24: Fan-out-of-four Load

The four inverters represent four separate loads, so the inputs must be connected. After the layout was finished, the fan-out-four (FO4) created above was placed on the output for simulation purposes. This can be seen in Figure 25.

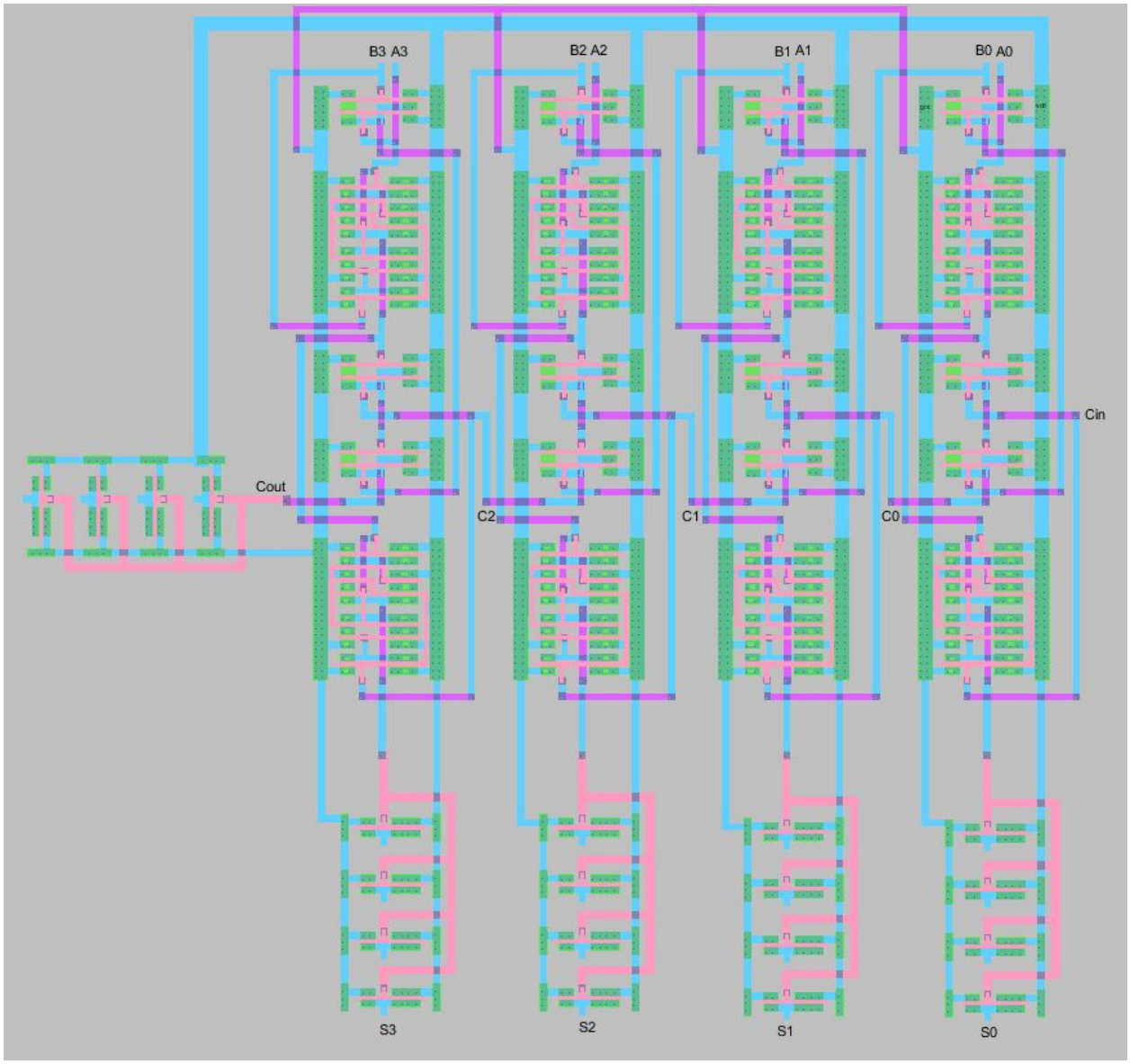

Figure 25: Four Bit Adder Layout with FO4's

2.23 Simulated Input / Output Waveforms for Four Bit Adder Layout

Finding the proper code for the simulation of the four bit adder layout was difficult because there are so many different input combinations possible. With 4 A inputs, 4 B inputs and a Cin input there are a total of 9 inputs. That means that there are $2^9 = 512$ possible

combinations. Because of the large amount of inputs, two simulations were performed and observed. Firstly, with Cin set to 0 for the duration and secondly Cin set to 1 for the entire duration. This way 256 possible input combinations were captured within each simulation. This makes it much easier to follow inputs and observe the outputs. The spice code used for the CMOS simulations can be seen in figure 26.

Case A (Cin is 0, observe sums and cout):	Case B (Cin is 1, observe sums and cout):
vdd vdd 0 DC 5	vdd vdd 0 DC 5
VCin Cin 0 DC 0	VCin Cin 0 DC 5
VA0 A0 0 pulse 5 0 0 2n 2n 256u 512u	VA0 A0 0 pulse 5 0 0 2n 2n 256u 512u
VB0 B0 0 pulse 5 0 0 2n 2n 128u 256u	VB0 B0 0 pulse 5 0 0 2n 2n 128u 256u
VA1 A1 0 pulse 5 0 0 2n 2n 64u 128u	VA1 A1 0 pulse 5 0 0 2n 2n 64u 128u
VB1 B1 0 pulse 5 0 0 2n 2n 32u 64u	VB1 B1 0 pulse 5 0 0 2n 2n 32u 64u
VA2 A2 0 pulse 5 0 0 2n 2n 16u 32u	VA2 A2 0 pulse 5 0 0 2n 2n 16u 32u
VB2 B2 0 pulse 5 0 0 2n 2n 8u 16u	VB2 B2 0 pulse 5 0 0 2n 2n 8u 16u
VA3 A3 0 pulse 5 0 0 2n 2n 4u 8u	VA3 A3 0 pulse 5 0 0 2n 2n 4u 8u
VB3 B3 0 pulse 5 0 0 2n 2n 2u 4u	VB3 B3 0 pulse 5 0 0 2n 2n 2u 4u
.tran 0 600u	.tran 0 600u
.include P:\RadhakrD\C5 models.txt	.include P:\RadhakrD\C5 models.txt

Figure 26: Spice code for Four Bit Adder Simulations

The first simulation with Cin set to 0 can be seen in figure 27.

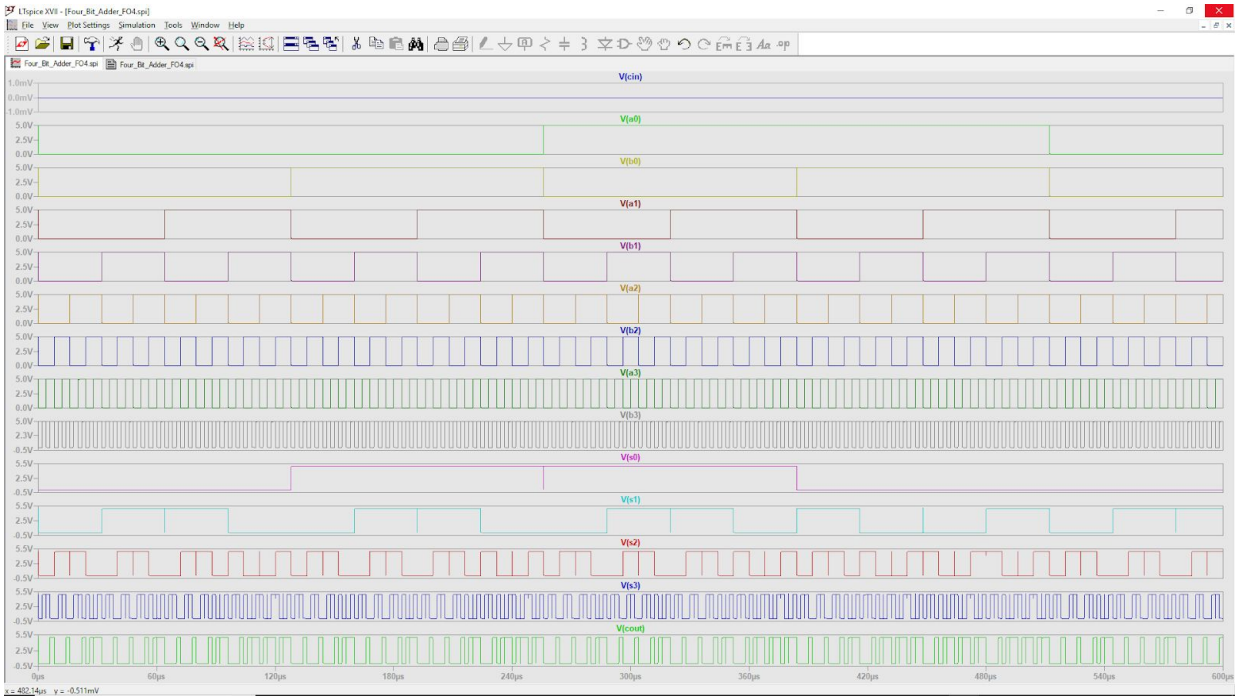

Figure 27: CMOS Simulation of Four Bit Adder with Cin set to 0

Secondly, the simulation of the four bit adder with Cin set to 1 can be seen in Figure 28.

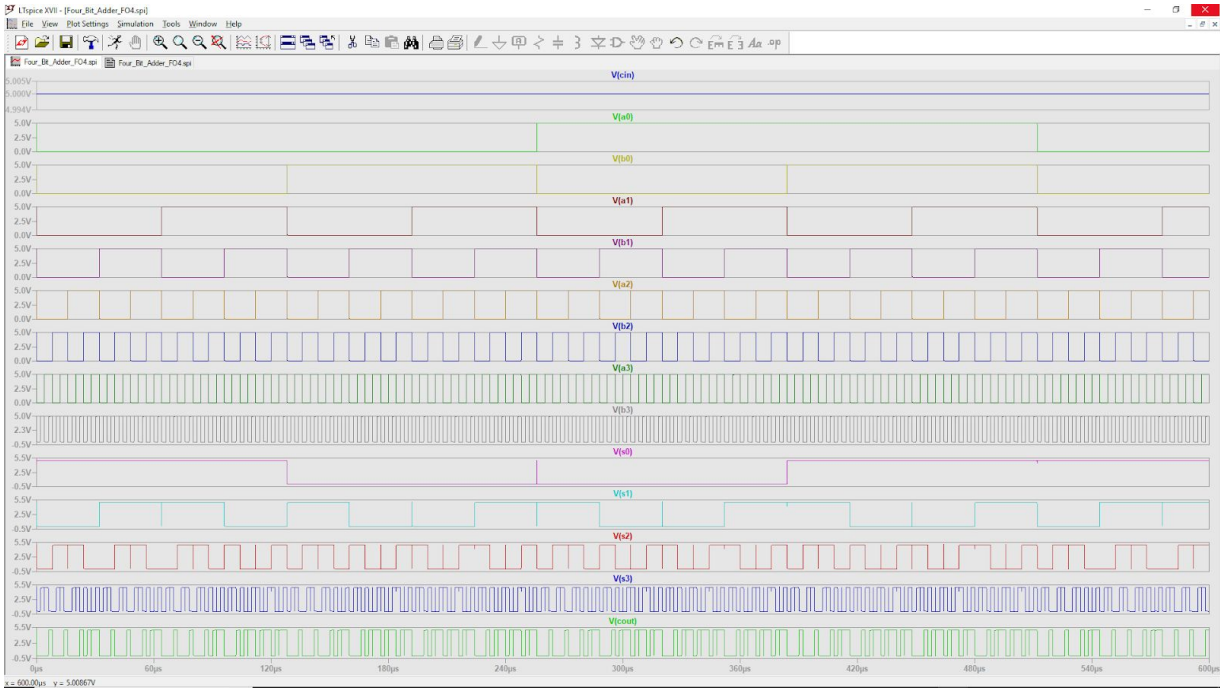

Figure 28: CMOS Simulation of Four Bit Adder with Cin set to 1

2.24 Four Bit Adder Icon

The four bit adder icon was created for simple implementation in future designs. The schematic and layout have an identical icon, which can be seen in figure 29. The icon has inputs A3-A0, B3-B0 and Cin labeled. It also has outputs Cout and S3-S0 labeled. Vdd and gnd are labeled as well.

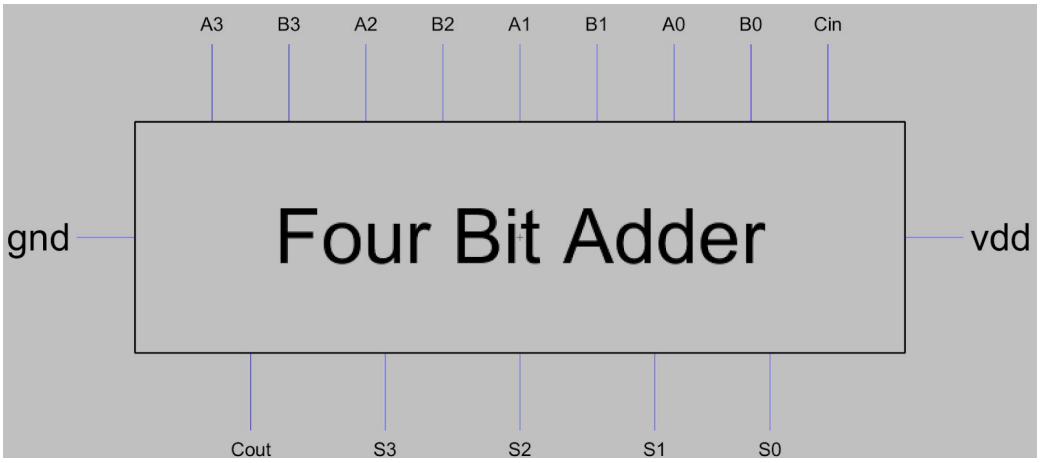

Figure 29: Four Bit Adder Icon

2.3 Four Bit Adder-Subtractor

The following sections detail how the four bit adder-subtractor was created and tested once the prior components were finalized. Once the four bit adder was finished, the four bit adder-subtractor could be easily designed by modifying a few of the inputs and outputs.

2.31 Transistor Schematic for Four Bit Adder-Subtractor

The “carry in” from the original four bit adder-subtractor became a “sub line” denoted “sub” to select the operation to be performed by the circuit. When sub was high logic, the system subtracted. The sub line was XOR’ed with each B input and fed to where the original B inputs were. The carry out of the final adder-subtractor module was inverted, and then could have been ANDed with the sub bit. This gave the sign output bit. In this design, NANDs were used, so using equivalent logic, the carry out was inverted, NANDed with the sub bit, and then inverted again. Lastly, the compliment of the sub bit was NANDed with carry out and then inverted. This logic gave our new carry output. The designed CMOS circuit can be seen in Figure 30.

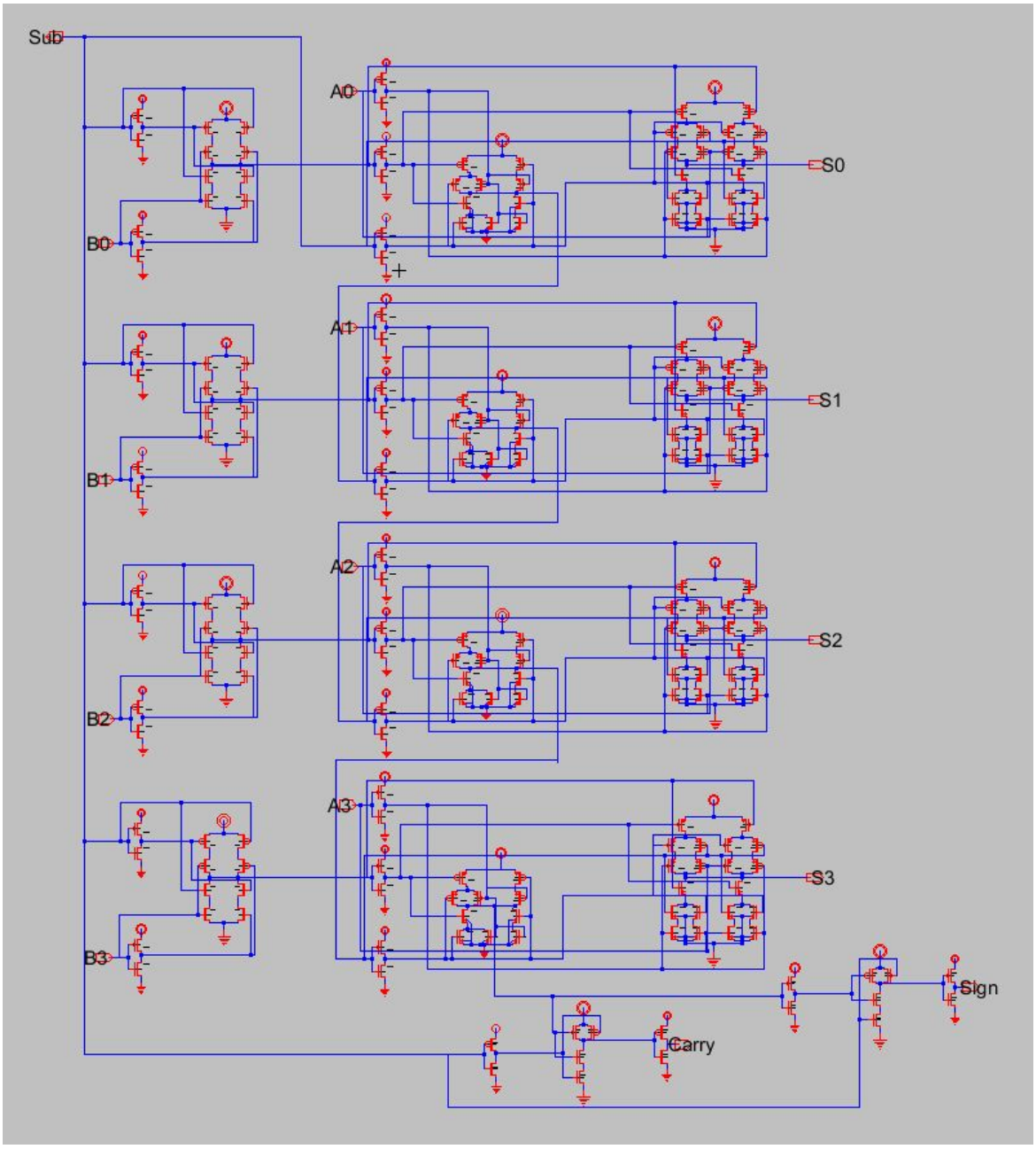

Figure 30: CMOS Schematic of Four Bit Adder-Subtractor

2.32 Layout Design of Four Bit Adder-Subtractor

For the CMOS schematic seen in Figure 30, there is no FO4 load. This is because Electric VLSI simulation software does not recognize the FO4 as a capacitance load in the schematics.

This is not the case for the layout however, and such the FO4 load was connected to each output node. Electric VLSI recognizes the FO4 load as a parasitic capacitance. The design for the layout closely resembles that of the four bit adder layout seen in Figure 25. The adder-subtractor can be seen in Figure 31.

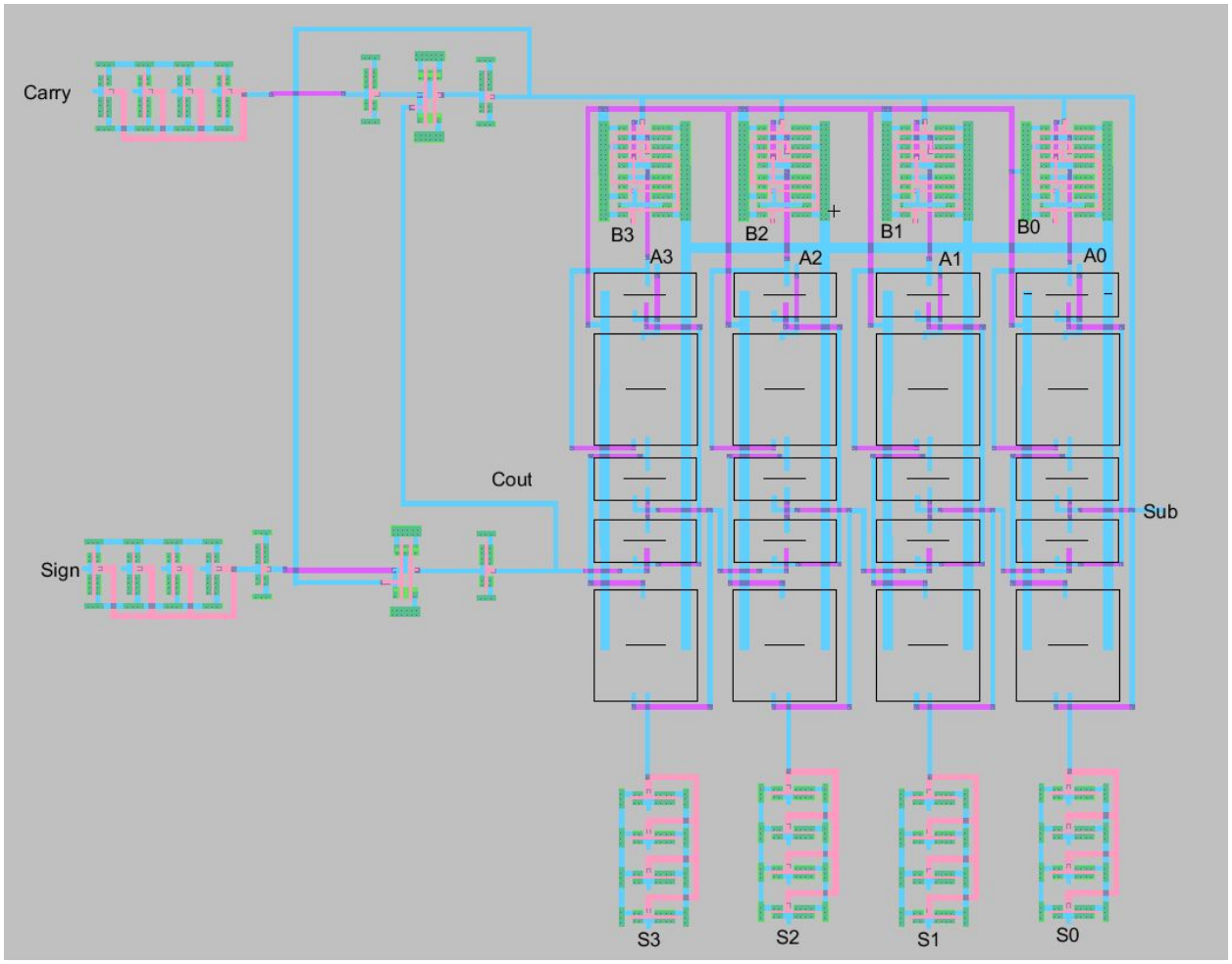

Figure 31: Four Bit Adder-Subtractor Layout

2.33 Padframe Layout

The following section outlines the creation of the padframe chip for the circuit. The reason for creating a padframe layout is to show the design in its final form that would be sent out for fabrication. Unfortunately there is no way to test the chip for errors, the only way would be to fabricate it and test it physically. The pad with labeled pins can be seen in Figure 32.

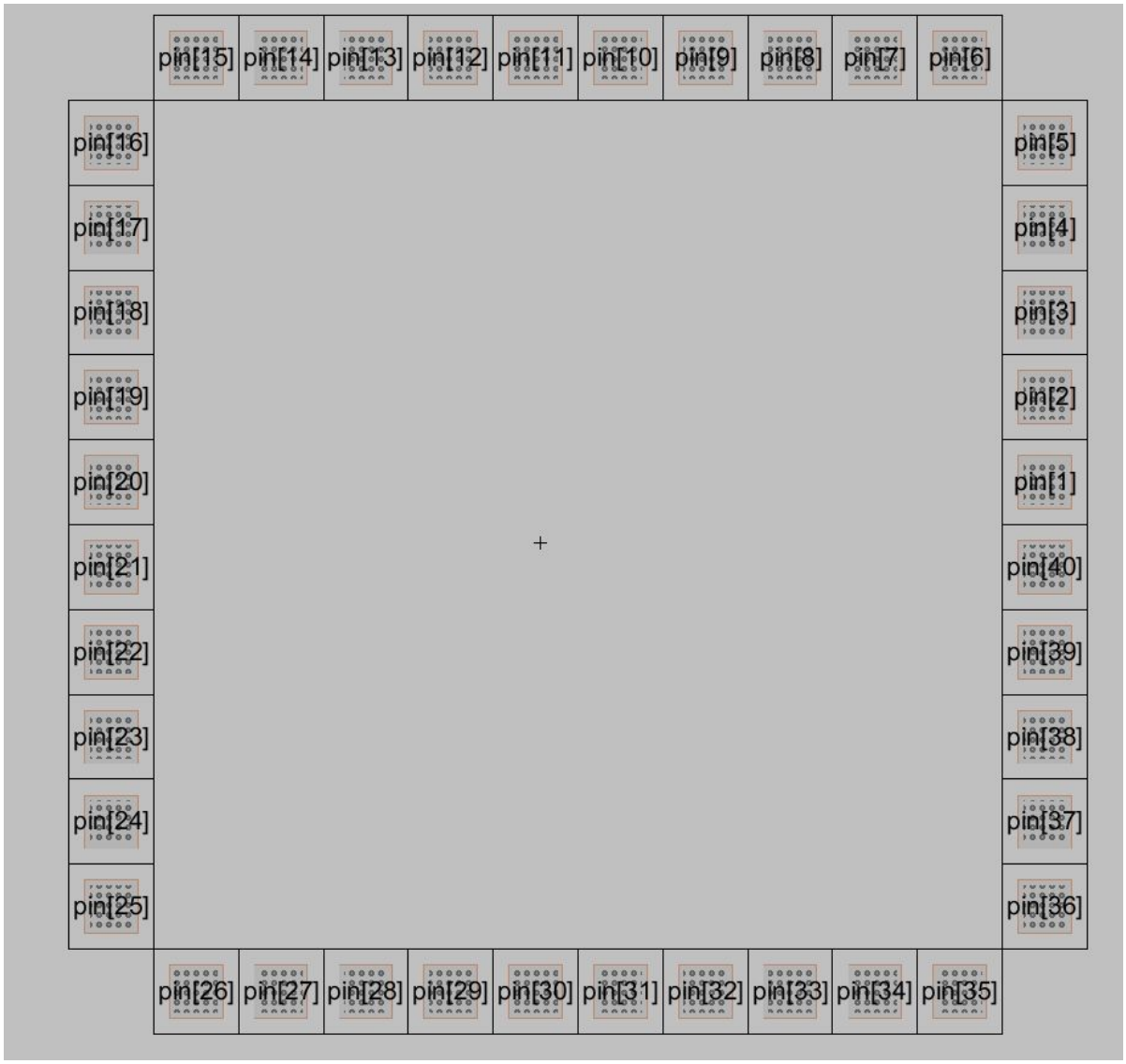

Figure 32: Pad with Pins Labeled

Once the pins were labeled, the icon was brought into a schematic view to name the pins. The pins on the icon were labeled in such a way that direct connections from the pad pins were easy to establish. The pin assignments can be seen in Figure 33.

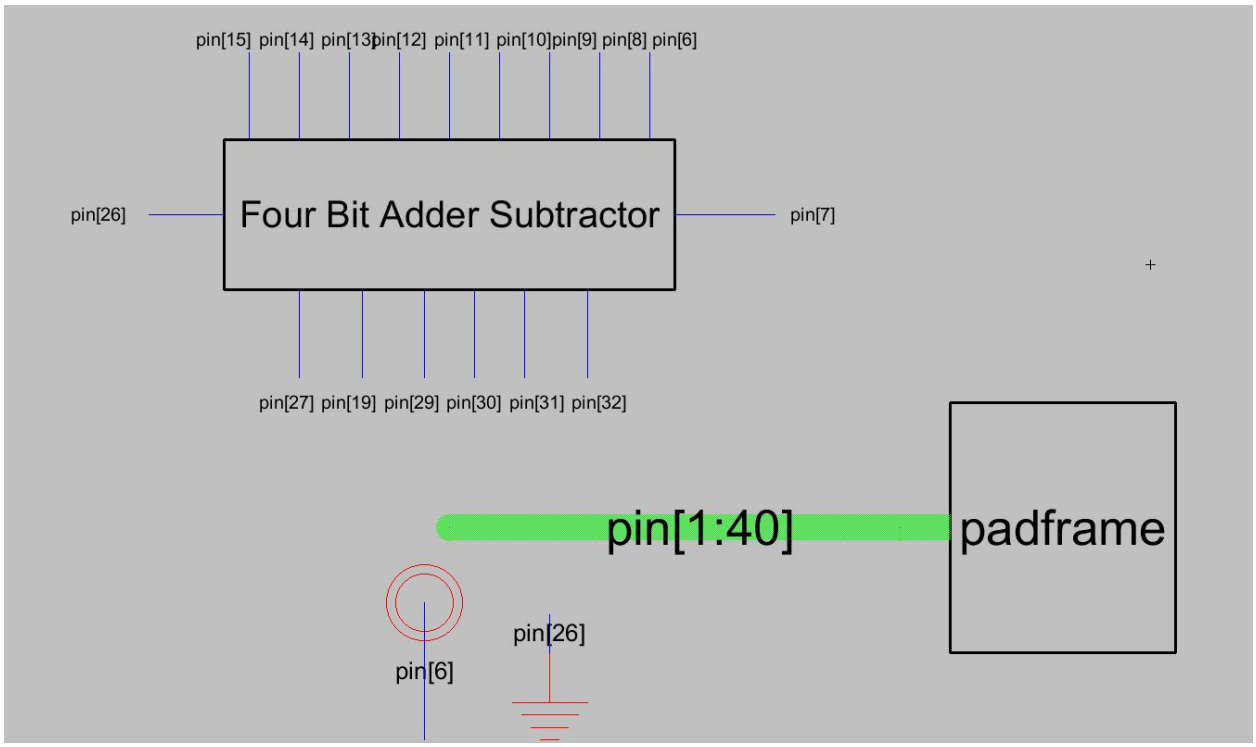

Figure 33: Four Bit Adder-Subtractor I/O Pad Pin Assignment

After the icon was created establishing the connections between the leads and I/O padframe, metal 3 was used to extend the exports of the layout. The layout is placed within the I/O padframe rather than the icon. The layout with leads extended can be seen in Figure 34.

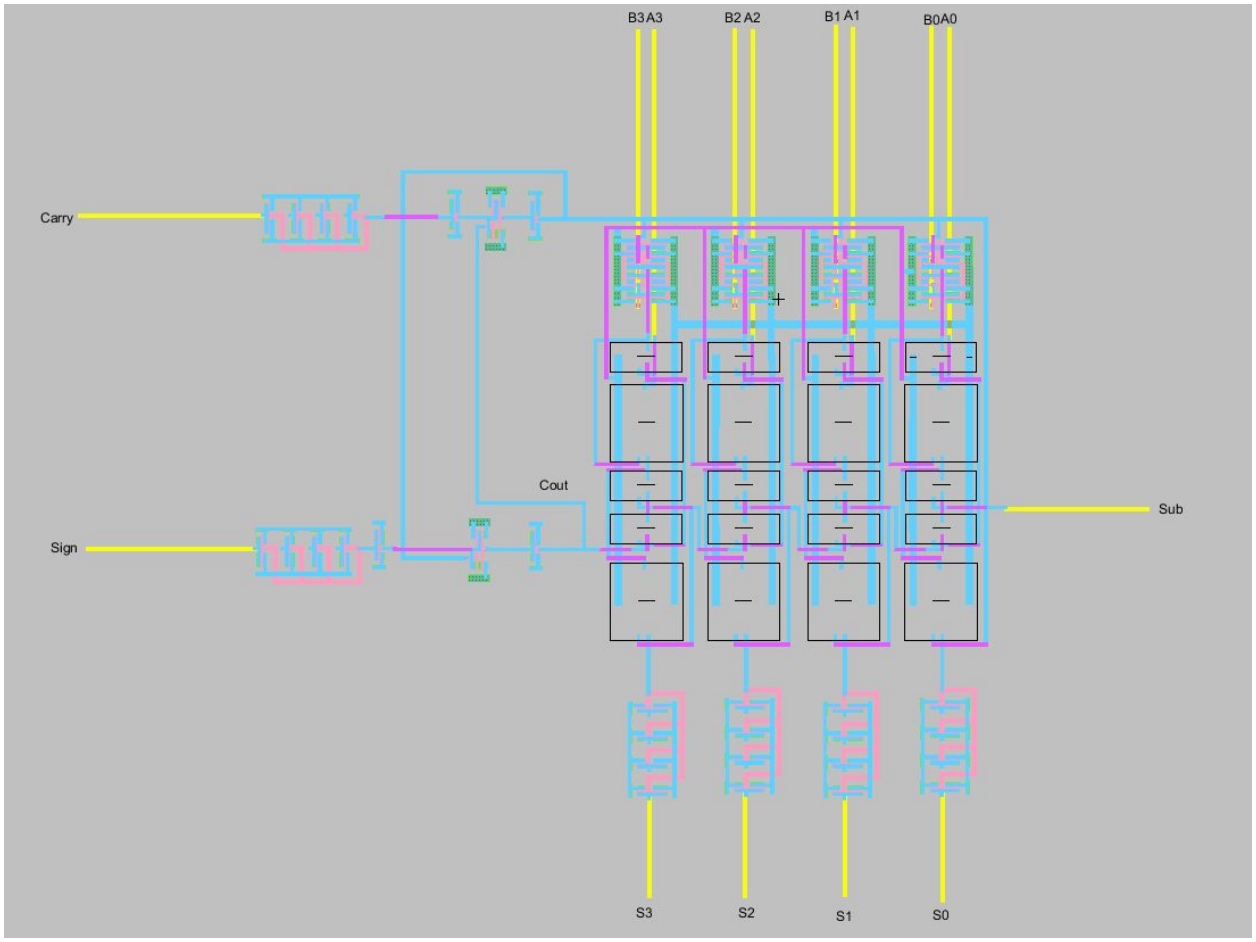

Figure 34: Four Bit Adder-Subtractor Layout with Leads Extended

Finally, the layout shown in Figure 34 was placed in the I/O padframe seen in Figure 32 using the pin assignments shown in Figure 33. The final four bit adder-subtractor padframe layout can be seen in Figure 35.

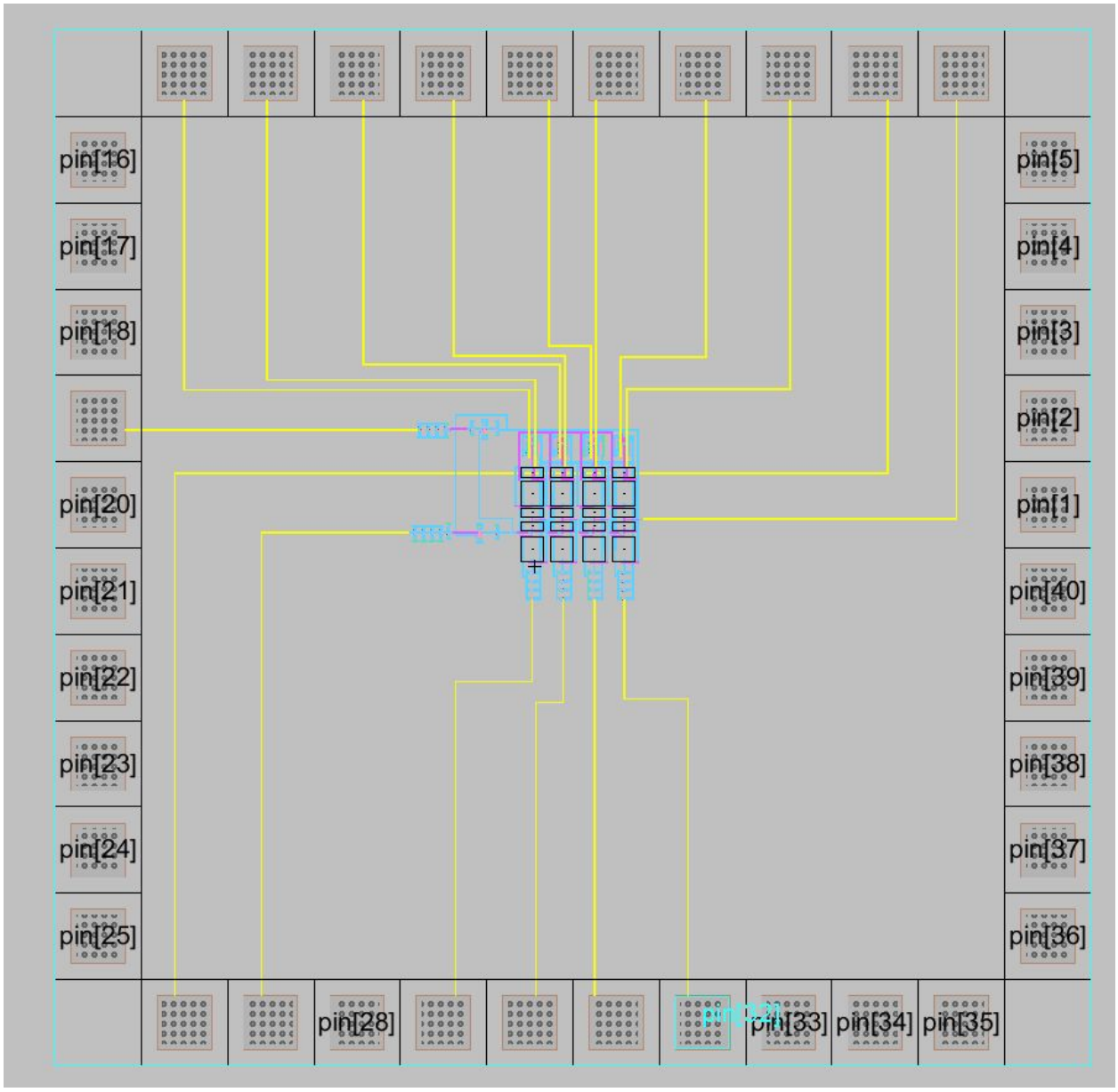

Figure 35: Four Bit Adder-Subtractor Padframe

3. Verification

The following section covers the simulation and verification of the behavior of the transistor schematics and layouts of the four bit adder-subtractor with the FO4 output load. The design was tested with four specific test cases in order to more easily understand the behavior. Propagation was also measured using spice code, as well as the FO4's effects on the propagation delay. Each test case took the two sets of inputs (4 A and 4 B), as well as the Sub input. The inputs were assigned DC voltage of either 0 or 5 volts and the sub line was pulsed to add and subtract the values every 40 ns.

The code used in the first test case is shown in Figure 36. The test case code uses A inputs 1111 and B inputs 0000. The sub bit is pulsed. Cout was monitored.

```
Case 1

vdd vdd 0 DC 5
vA0 A0 0 DC 5
vA1 A1 0 DC 5
vA2 A2 0 DC 5
vA3 A3 0 DC 5
vB0 B0 0 DC 0
vB1 B1 0 DC 0
vB2 B2 0 DC 0
vB3 B3 0 DC 0
vSub Sub 0 pulse (0 5 0 .01ns .01ns 20ns 40ns)
.tran 0 100n
.include C:\Users\walby\C5_models.txt
```

Figure 36: Four Bit Adder-Subtractor Spice Code for Test Case 1

The output simulation waveform for the first test case can be seen in Figure 37.

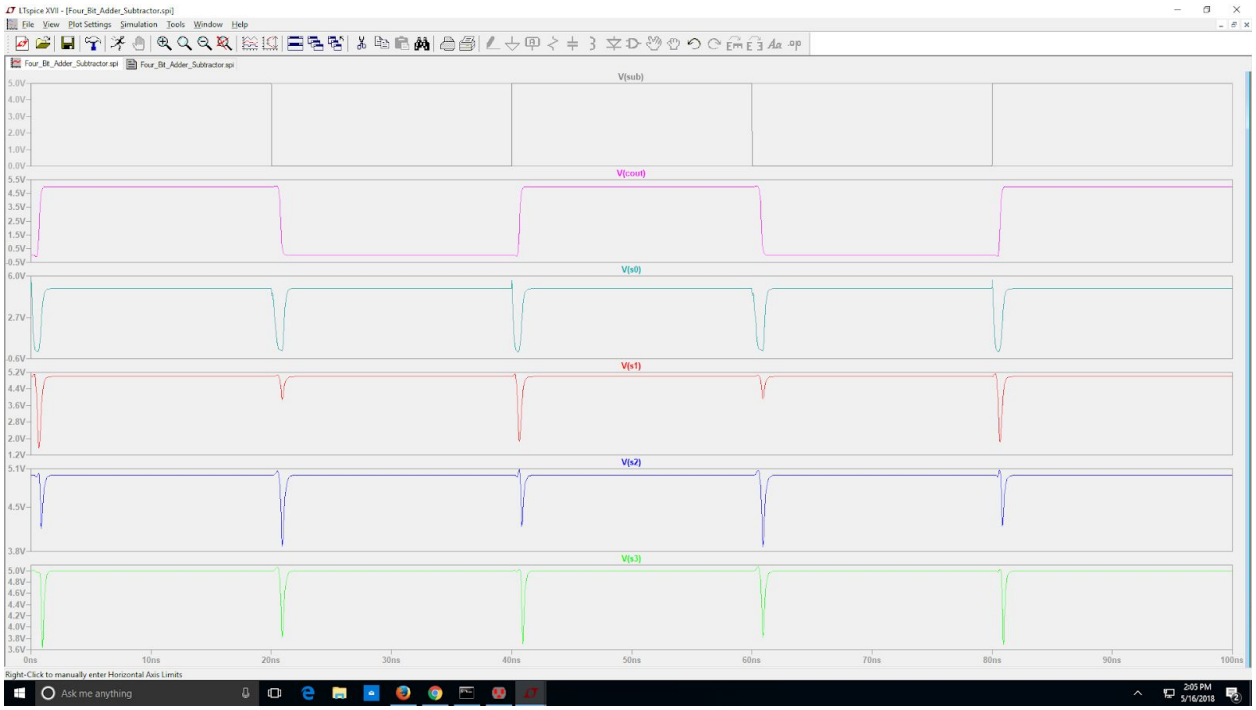

Figure 37: Four Bit Adder-Subtractor Test 1

The code used in the second test case is shown in Figure 38. The test case code uses A inputs 1111 and B inputs 0000. The sub bit is pulsed. The reason for testing this way was to monitor the cout value when it always equaled 1.

```
Case 2

vdd vdd 0 DC 5
vA0 A0 0 DC 5
vA1 A1 0 DC 5
vA2 A2 0 DC 5
vA3 A3 0 DC 5
vB0 B0 0 DC 5
vB1 B1 0 DC 0
vB2 B2 0 DC 0
vB3 B3 0 DC 0
vSub Sub 0 pulse (0 5 0 .01ns .01ns 20ns 40ns)
.tran 0 100n
.include C:\Users\walby\C5_models.txt
```

Figure 38: Four Bit Adder-Subtractor Spice Code for Test Case 2

The output simulation waveform for the second test case can be seen in Figure 39.

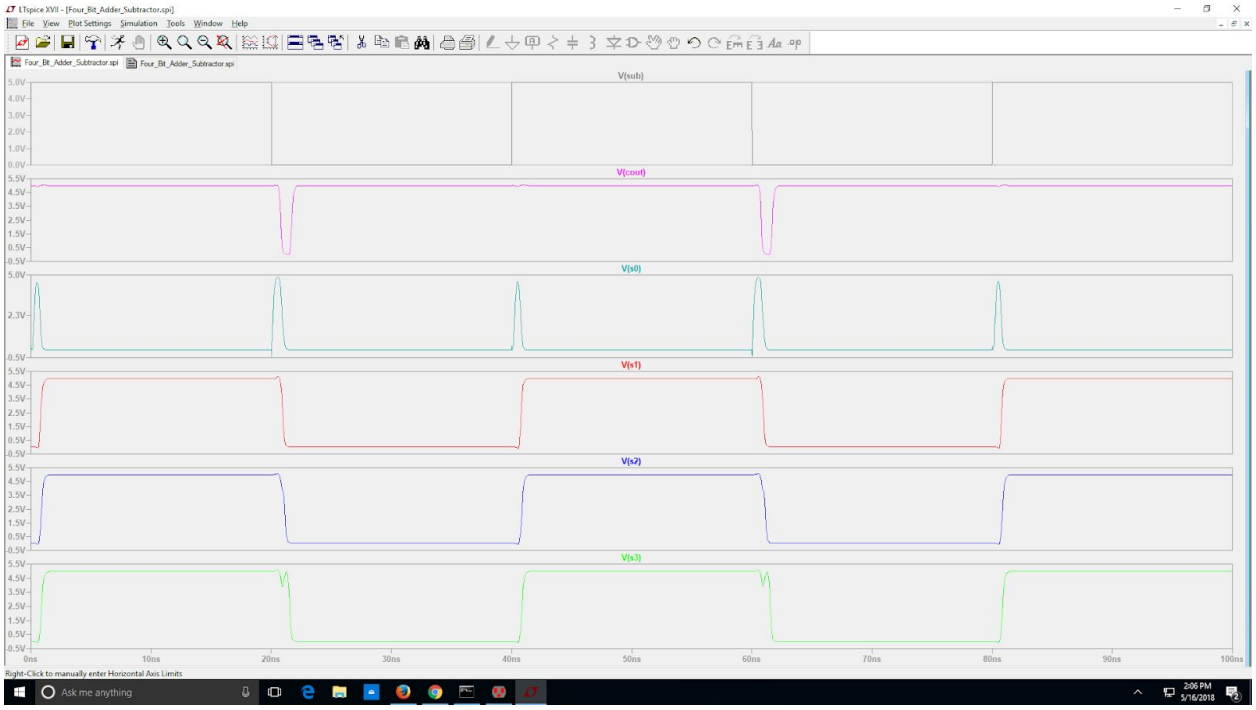

Figure 39: Four Bit Adder-Subtractor Test 2

The code used in the third test case is shown in Figure 40. The test case code uses A inputs 0001 and B inputs 1111. The sub bit is pulsed. The reason for testing this way was to monitor how the cout value acted when toggling the opposite of the first test case.

Case 3		
vdd vdd 0 DC 5		
vA0 A0 0 DC 5		
vA1 A1 0 DC 0		
vA2 A2 0 DC 0		
vA3 A3 0 DC 0		
vB0 B0 0 DC 5		
vB1 B1 0 DC 5		
vB2 B2 0 DC 5		
vB3 B3 0 DC 5		
vSub Sub 0 pulse (0 5 0 .01ns .01ns 20ns 40ns)		
.tran 0 100n		
.include C:\Users\walby\C5_models.txt		

Figure 40: Four Bit Adder-Subtractor Spice Code for Test Case 3

The output simulation waveform for the third test case can be seen in Figure 41.

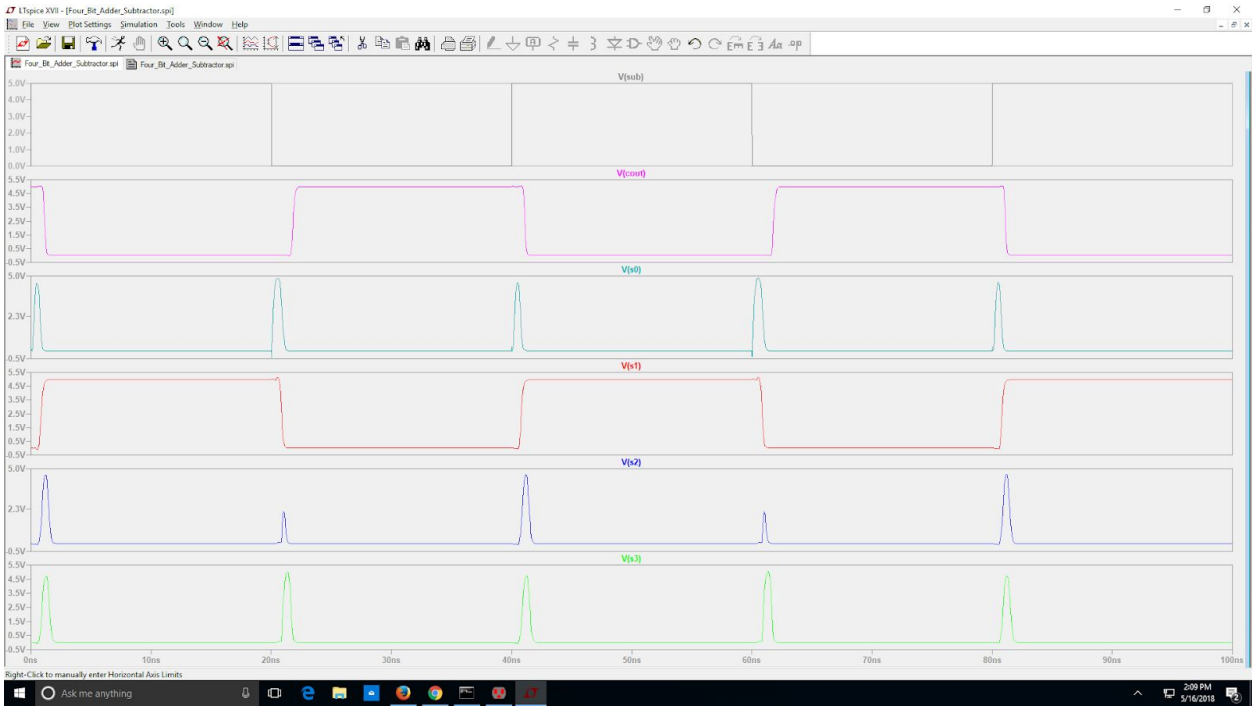

Figure 41: Four Bit Adder-Subtractor Test 3

The code used in the fourth test case is shown in Figure 42. The test case code uses A inputs 1000 and B inputs 1000. The sub bit is pulsed. The reason for testing this way was to monitor the cout value when it had a constant value of 1. Sum 3 to sum 0 values all read 0.

```
Case 4

vdd vdd 0 DC 5
vA0 A0 0 DC 0
vA1 A1 0 DC 0
vA2 A2 0 DC 0
vA3 A3 0 DC 5
vB0 B0 0 DC 0
vB1 B1 0 DC 0
vB2 B2 0 DC 0
vB3 B3 0 DC 5
vSub Sub 0 pulse (0 5 0 .01ns .01ns 20ns 40ns)
.tran 0 100n
.include C:\Users\walby\C5_models.txt
```

Figure 42: Four Bit Adder-Subtractor Spice Code for Test Case 4

The output simulation waveform for the fourth test case can be seen in Figure 43.

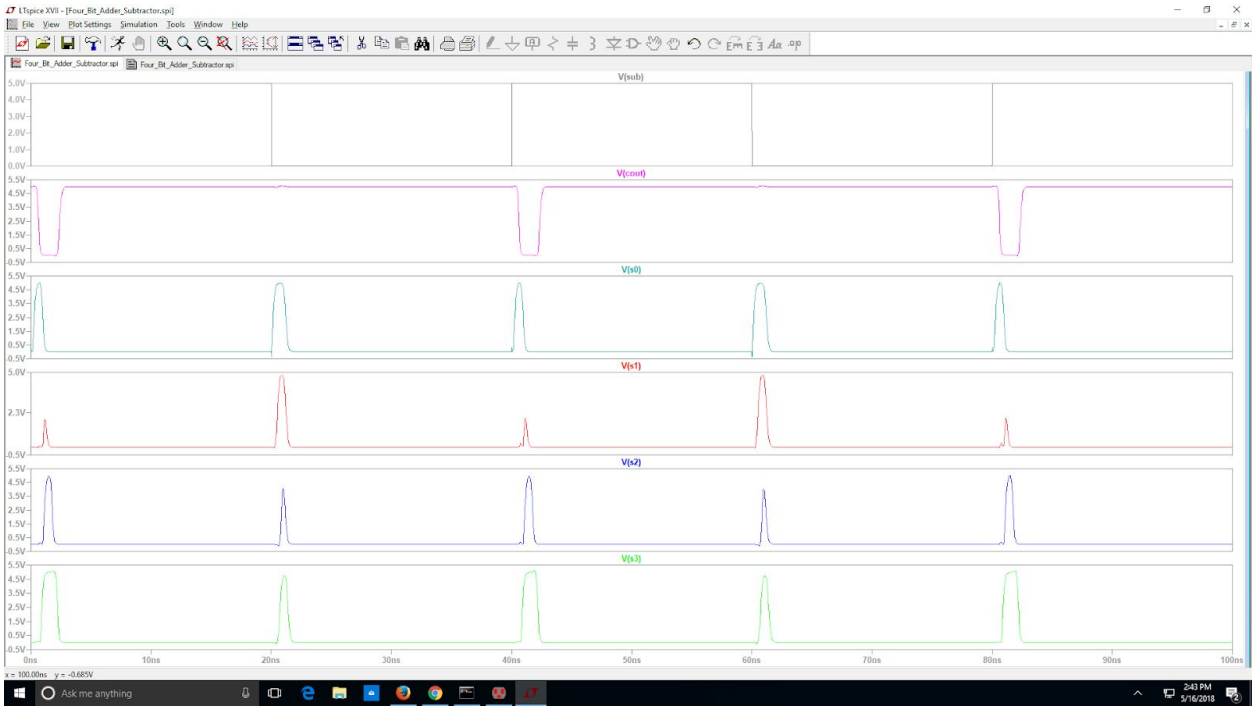

Figure 43: Four Bit Adder-Subtractor Test 4

4. Measurements & Analysis

This section will include the propagation delay analysis that the FO4 gives to the four bit adder-subtractor layout. Propagation delays, rise times, fall times, and speed of the four bit adder-subtractor circuit were all calculated and listed in this section. To find the most accurate values for the delay, spice code was added specifically to test rise and fall times.

The code used to find the propagation delay can be seen in Figure 44.

```
Propagation Delay

vdd vdd 0 DC 5
vA0 A0 0 DC 5
vA1 A1 0 DC 5
vA2 A2 0 DC 5
vA3 A3 0 DC 5
vB0 B0 0 DC 0
vB1 B1 0 DC 0
vB2 B2 0 DC 0
vB3 B3 0 DC 0
vSub Sub 0 pulse (0 5 0 .01ns .01ns 20ns 40ns)
.meas tran tdelay TRIG v(Sub) val=2.5 RISE=2 TARG V(Carry) val=2.5 RISE=2
.meas tran tdelay2 TRIG v(Sub) val=2.5 FALL=2 TARG V(Carry) val=2.5 FALL=2
.tran 0 100n
.include C:\Users\walby\C5_models.txt
```

Figure 44: Four Bit Adder-Subtractor Spice Code for Propagation Delay

The two lines of code are similar, although one is for rise time and one is for fall time. Tdelay refers to the time propagation delay from low to high by looking at the rising edge. The reference point used is named trig and the output value is listed as targ. The numbers are then 50% past in the low to high or high to low distance. This is shown in the ‘val = 2.5’ bit. The second propagation delay line is for the falling edge of the waveform.

Cout was monitored because it has the largest propagation delay, due to being at the end of the four adder-subtractor modules. The total delay for the value is each individual delay added together. The measurements were found and can be seen in Figure 45.

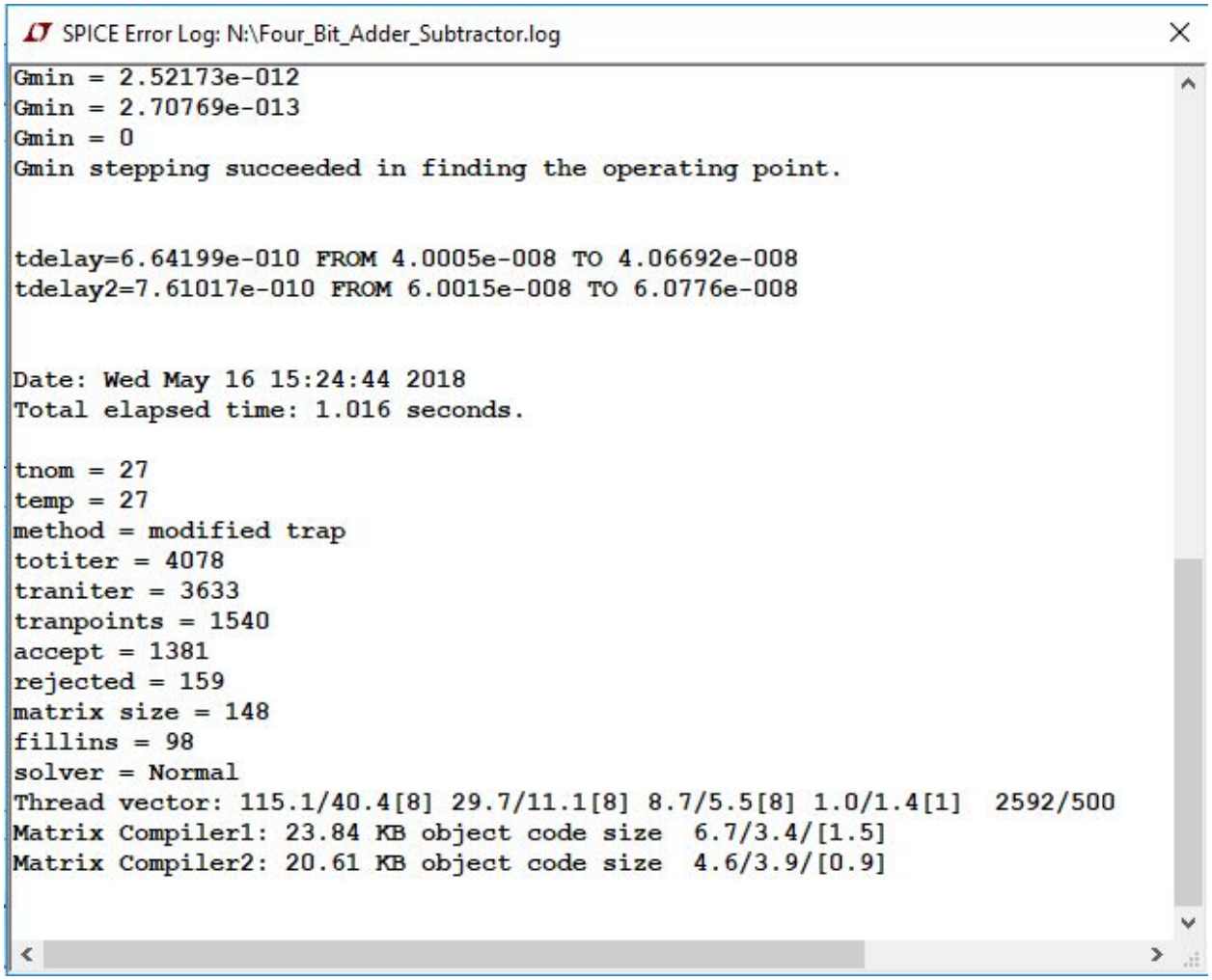

Figure 45: SPICE Error Log for Propagation Delay

The times tdelay and tdelay2 were plugged into the following equation to find the average propagation delay:

$$\frac{T_{PLH}+T_{PHL}}{2}$$

$$= \frac{6.64199*10^{-10}+7.61017*10^{-10}}{2}$$

$$= \frac{1.425216*10^{-9}}{2}$$

$$= 0.712608 * 10^{-9}\ s\ = 0.712608\ ns$$

The measurements for rise time and fall time are shown with more detail in Figures 46 and 47 respectively. The two times were taken at 10% and 90% of their respective slopes

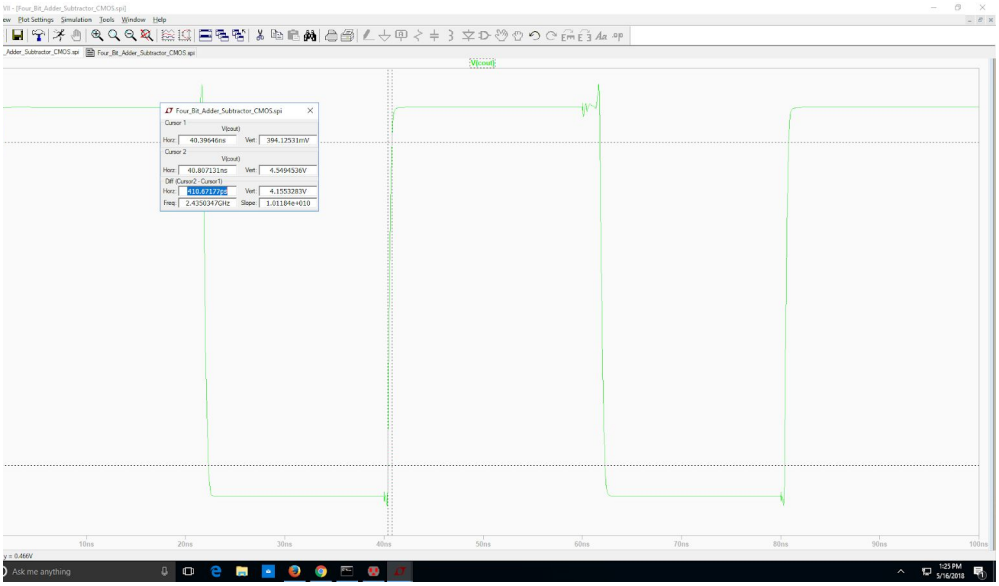

Figure 46: Measurement for Rise Time

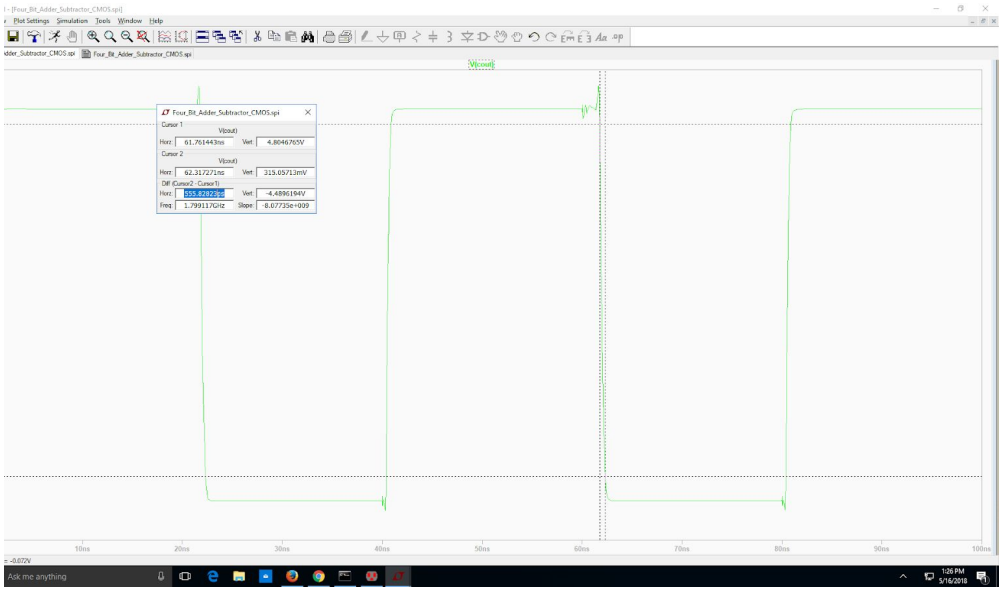

Figure 47: Measurement for Fall Time

The values found for rise and fall time are highlighted in Figures 48 and 49.

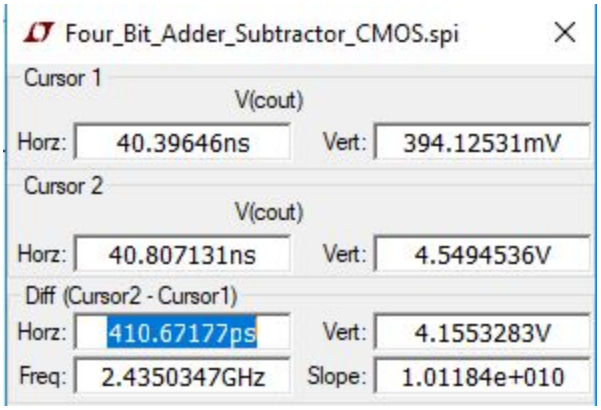

Figure 48: Rise Time Window

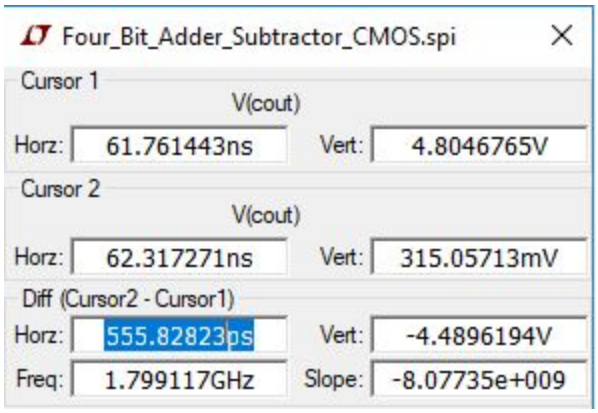

Figure 49: Fall Time Window

Lastly, the speed of the circuit was analyzed. To do this, the cout output was needed to rise before the cin input reached the falling edge. The period needed to be found first and from then the frequency could be calculated. The code used for this can be seen in Figure 50.

```
vdd vdd 0 DC 5
vA0 A0 0 DC 5
vA1 A1 0 DC 5
vA2 A2 0 DC 5
vA3 A3 0 DC 5
vB0 B0 0 DC 0
vB1 B1 0 DC 0
vB2 B2 0 DC 0
vB3 B3 0 DC 0
vSub Sub 0 pulse (0 5 0 .01ns .01ns 1ns 3ns)
.tran 0 100n
.include P:\RadhakrD\C5_models.txt
```

Figure 50: Code for Speed Analysis

The accompanying waveform for the spice code can be seen in Figure 51.

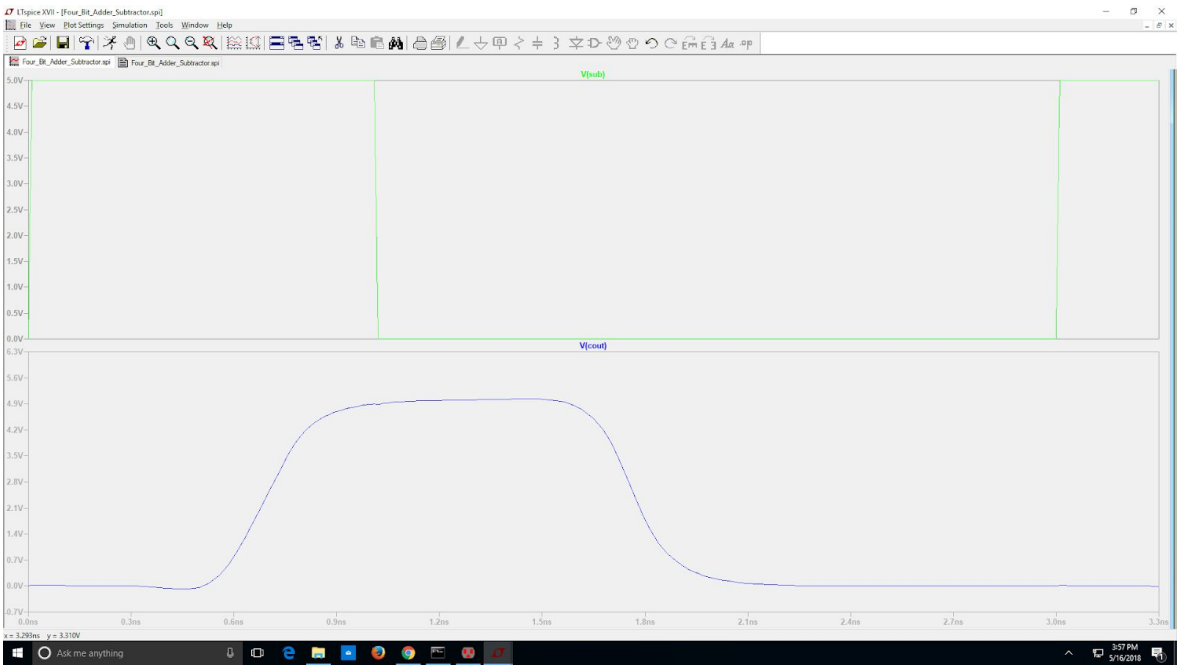

Figure 51: Waveform Output for Speed Analysis

Observing the waveform output, the period was found to be 3ns. From there the frequency was calculated:

$$f = \frac{1}{T} = \frac{1}{3*10^{-9}s} = \frac{1*10^{9}}{3s} = 333.3333\ MHz$$

5. Conclusion

This project solidified students' knowledge with the basics of the Electric VLSI tools and software by requiring and simulation and design of a four bit adder-subtractor. Through the use of truth tables, KMAPs and logic expressions, created with prior knowledge of digital logic and VLSI design, transistor schematics, layouts and icons were all able to be created. The project contained information on how to create the designs for a full adder, XOR and NAND gates, a fan-out-four and how to put these components together to create a complex design, the four bit adder-subtractor. New spice code was used for simulation, which was initially difficult to understand but later became easy to understand and implement for testing purposes. These commands were used to find propagation delay and to confirm boolean logic expressions and circuit behavior. Finally, a pad layout was created for the first time, which was designed as a chip which would be sent in ready to be fabricated.

Some problems encountered while completing this project were connecting the parts together without increasing the metal levels needed. Another issue was the fact that the device containing the libraries for some of the components was stolen during lab time. This set the

project back two weeks. A third problem was that towards the end of the design stage, the Electric VLSI software refused to save, resulting in a loss of another weeks worth of work.

Overall, the final design and project performed exactly as expected, as a four-bit adder subtractor was created with 9 inputs: A3-A0, B3-B0 and Sub. The four bit adder-subtractor also has 6 outputs: Sum3-Sum0, Carry Out and a Sign bit. Design Rule Check (DRC), Electrical Rule Check (ERC) and Network Consistency Check (NCC) were used periodically during development to confirm that the design followed all design rules and had proper connections.

For more information, refer to CMOSedu Tutorial 5 [3], isweb.redwoods.edu's description of a full adder [4] and wikimedia.org's full adder article [5].

Bibliography

[1] electronics-tutorials.ws. *Binary Adder*. [online] Available at: https://www.electronics-tutorials.ws/combination/comb_7.html [Accessed 26 Apr. 2018]

[2] electronics-course.com *Karnaugh Map.* [online] Available at: http://electronics-course.com/karnaugh-map [Accessed 14 Mar. 2018]

[3] cmosedu.com *CMOSedu Tutorial 5 – Design, layout, and simulation of a ring oscillator.* [online] Available at: http://cmosedu.com/videos/electric/tutorial5/electric_tutorial_5.htm [Accessed 1 May , 2018]

[4] isweb.redwoods.edu. *Full Adder*. [online] Available at: http://isweb.redwoods.edu/instruct/calderwoodd/diglogic/full.htm [Accessed 14 Mar. 2018]

[5] commons.wikimedia.org. *Full Adder Blocks*. [online] Available at: https://commons.wikimedia.org/wiki/File:Full_Adder_Blocks.svg [Accessed 14 Mar. 2018]