EE457: Digital IC Design

Project #2

Report Cover Sheet

Due 3/21/2020

***PROJECT TITLE:___8 bit Ripple Carry Adder_____**

*Student Name:_____Aninda Halder_____

*Do not hand-write.

Sections (**Do not change order of sections**.)	GRADE Points
1. Executive Summary (1/2-page)	/0
2. Introduction (2 pages)	/5
3. Electric Circuit Schematic(transistor level)	/10
4. LTSpice simulations of Schematic (label waveforms. Verifications)	/10
5. IRSIM simulations of Schematic (label waveforms. Verifications)	/10
6. Electric Layout (show all details)	/25
7. LTSpice simulation of Layout (verifications)	/10
8. IRSIM Simulations of layout (verifications)	/10
9. Summary of Measurements in a) Propagation delays of critical path, b) Total Chip area in um^2	 /6 /4
10. Comparisons of Schematic & Layout (Find out any difference of delays)	/5
11. Conclusion	/5
TOTAL	**/100**

Table of Contents

Section 1: Executive Summary

For the project 2 of Spring 2020 class of EE457, I will design a CMOS of 8-Bit Ripple Carry Adder. To design the RCA, I will be using Electric to construct two different design of the RCA. First, I will design the schematic and then the Layout. After the design we will perform two different simulation, LTSPICE and IRSIM, to generate waveform to validate the outputs with expected outputs in order to validate our design. I will also use the generated waveforms to compare and analyze the differences between the Schematic and Layout designs.

To construct the design for the 8-Bit Ripple Carry Adder I will first design a Full adder. My Full Adder will be comprised of 2 XOR gates and 3 NAND gates. After designing the Full Adder, we will make sure it passes all the DRC check and Well check. I will also perform a LTPICE simulation to make sure the Full Adder performs expectedly. After the completion of the Full adder I will start the design of the 8-bit Ripple Carry Adder by Cascading 8 single Full Adders. The Full Adders will include three inputs, and two outputs, One for the Sum and another for the carryout which will be fed into the next Full Adder's Carry in input. If my logic works correctly, we will obtain a Ripple carry Adder. After the Ripple Carry Adder deign passes all the DRC checks I will move on to generate waveforms in LTSPICE and IRSIM to validate whether the design passes all the requirements. This will give me the opportunity to analyze and troubleshoot my design if needed and also allow me to improve the performance of the design.

Section 2: Introduction

The purpose of the a 8-Bit Ripple Carry Adder is to generate the sum of two 8-Bit binary numbers. The arithmetic addition can be expressed as the following Boolean expression F = A+B, where the Ripple Carry Adder would add individual bits from the least to most significant bit and will carry the overflow forward. To construct the design for the 8-Bit Ripple Carry Adder I will first design a Full adder. My Full Adder will be comprised of 2 XOR gates and 3 NAND gates. I will first build the Full Adder as it will be more realistic by building the individual units first. After designing the Full Adder, we will make sure it passes all the DRC check and Well check. I will also perform a LTPICE simulation to make sure the Full Adder performs expectedly. After the completion of the Full adder I will start the design of the 8-bit Ripple Carry Adder by Cascading 8 single Full Adders. The Full Adders will include three inputs, and two outputs, One for the Sum and another for the carryout which will be fed into the next Full Adder's Carry in input.

I began to design a NAND gate first for our Full Adder. The Boolean Expression of a NAND gate is F =(A.B). The truth table of a Two input NAND gate is shown below

Table 1: Two Input NAND Gate Truth Table

Input: A	**Input: B**	**Out: A NAND B**
0	0	1
0	1	1
1	0	1
1	1	0

Below is the Electric Schematic of the two input NAND Gate

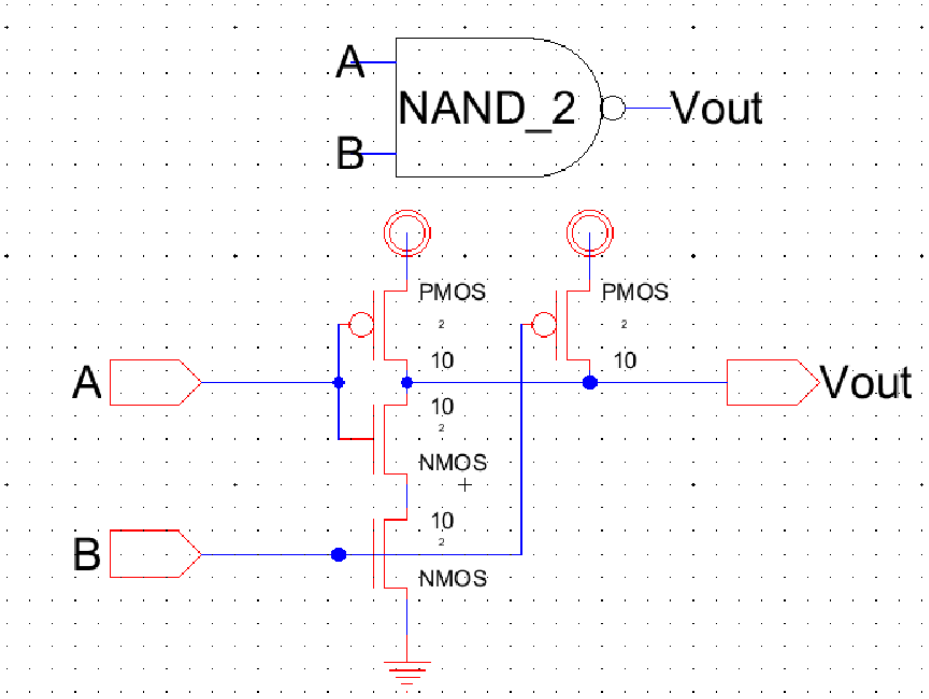

Figure 1: Schematic of Two Input NAND Gate

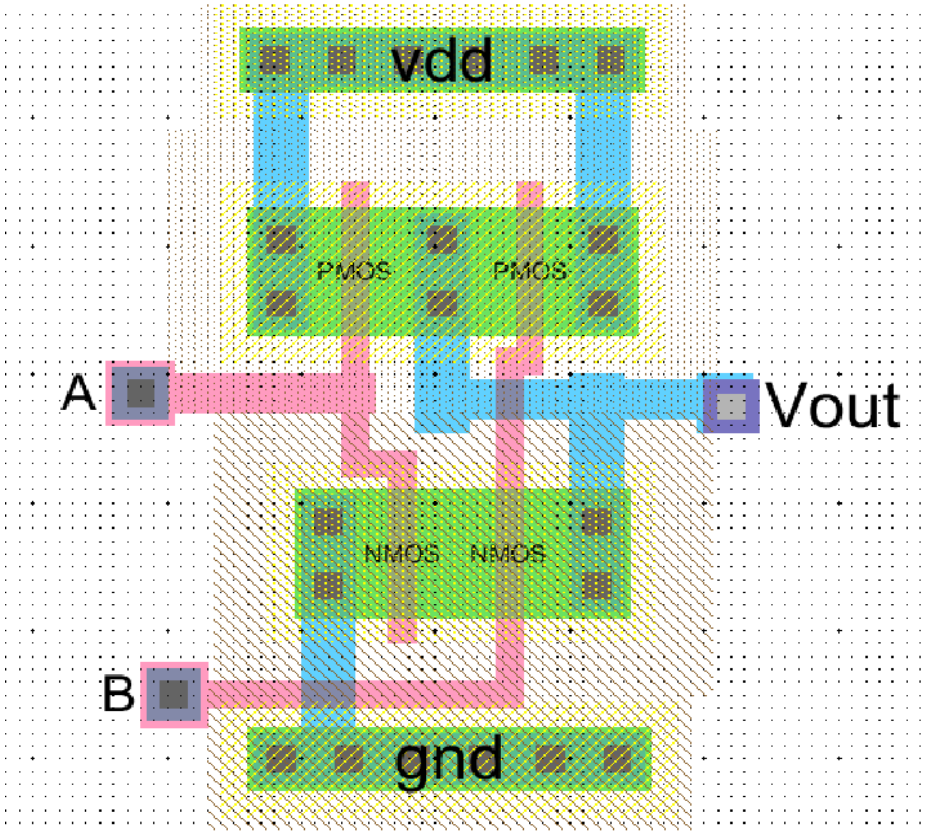

Figure 2: Layout of Two Input NAND Gate

Below is the truth table of the XOR gate

Table 2: Two Input XOR gate Truth Table

Input: A	**Input: B**	**Output: A XOR B**
0	0	0
0	1	1
1	0	1
1	1	0

Below is the Schematic of the XOR gate

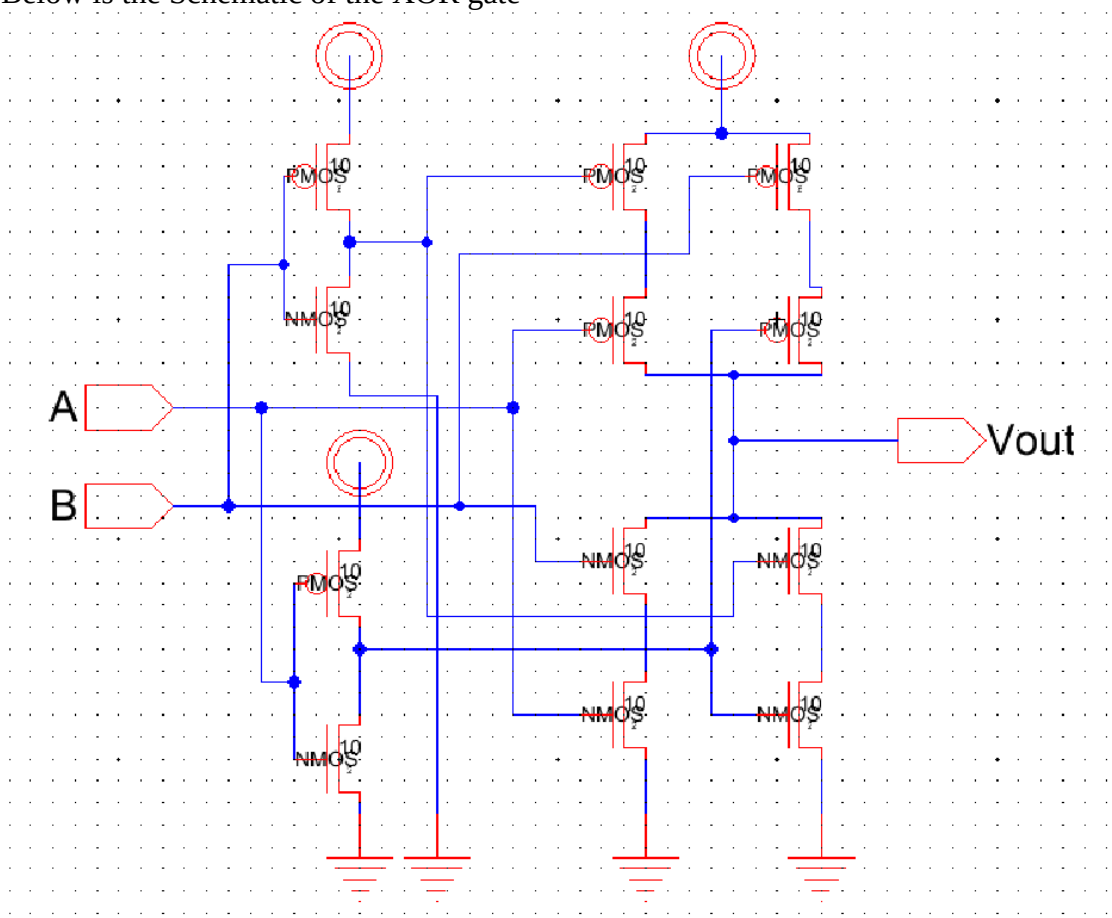

Figure 3: Schematic of Two Input XOR gate

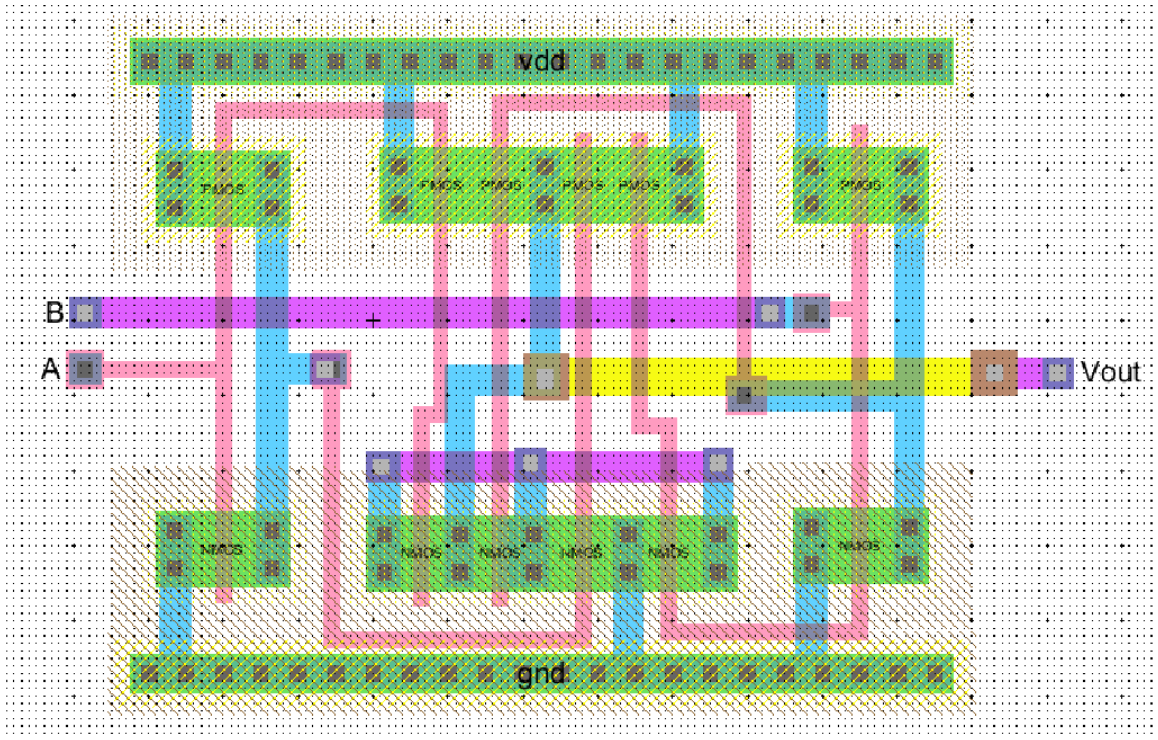

Figure 4: Layout of Two Input XOR gate

After I built the schematic and Layout of the XOR and NAND gate, I then constructed the schematic of the Full Adder. I will also run simulations of the Complete full Adder once its done and compare it to the Logical truth table for a Full Adder stated below.

Table 3: Truth Table for the Full Adder

Input: Cin	Input: A	Input: B	Output: Sum	Output: 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

Upon observing the Truth Table, we can see the addition process of two binary bits. The addition of 1 and 1 bit causes a overflow of bit 1, which has to stream outwards from Output Carry Out which will be Carried into the input of the next Full Adder thru the Input Carry In. Here is a Overview of the Logic.

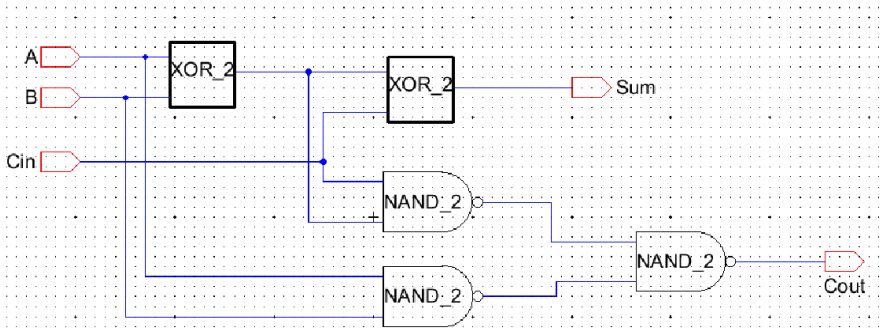

Figure 5: Overview of Schematic of Full Adder

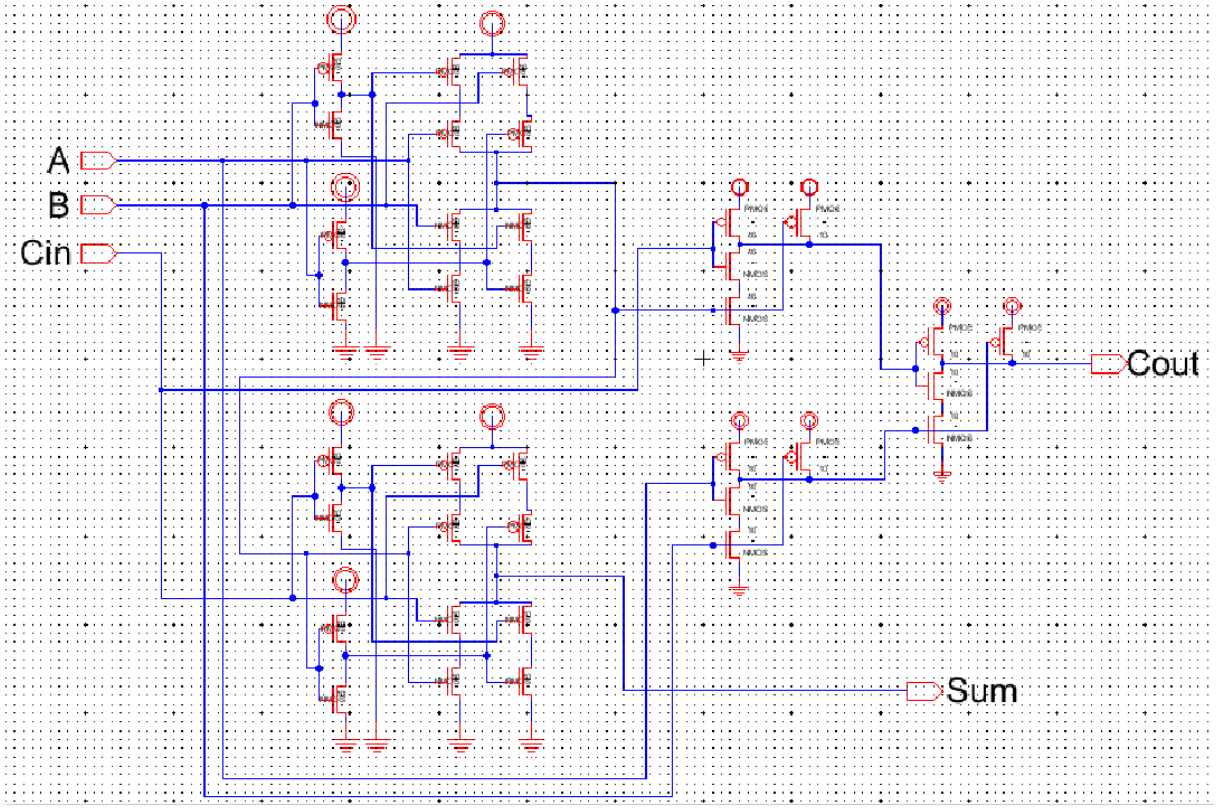

Figure 6:Schematic of Full Adder

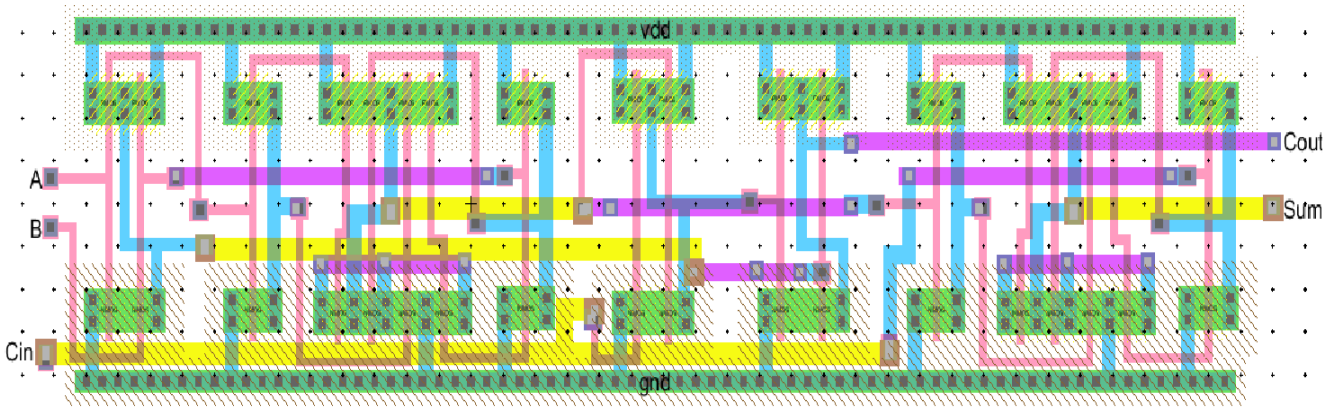

Figure 7: An Overview of the Layout of the Full Adder

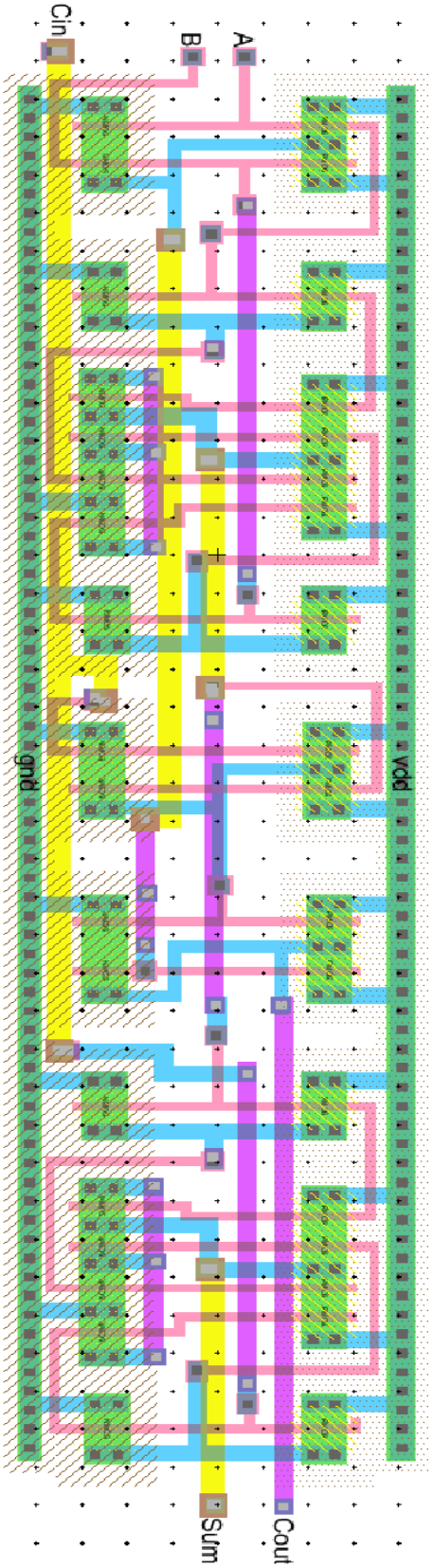

Figure 8: Layout of Full Adder

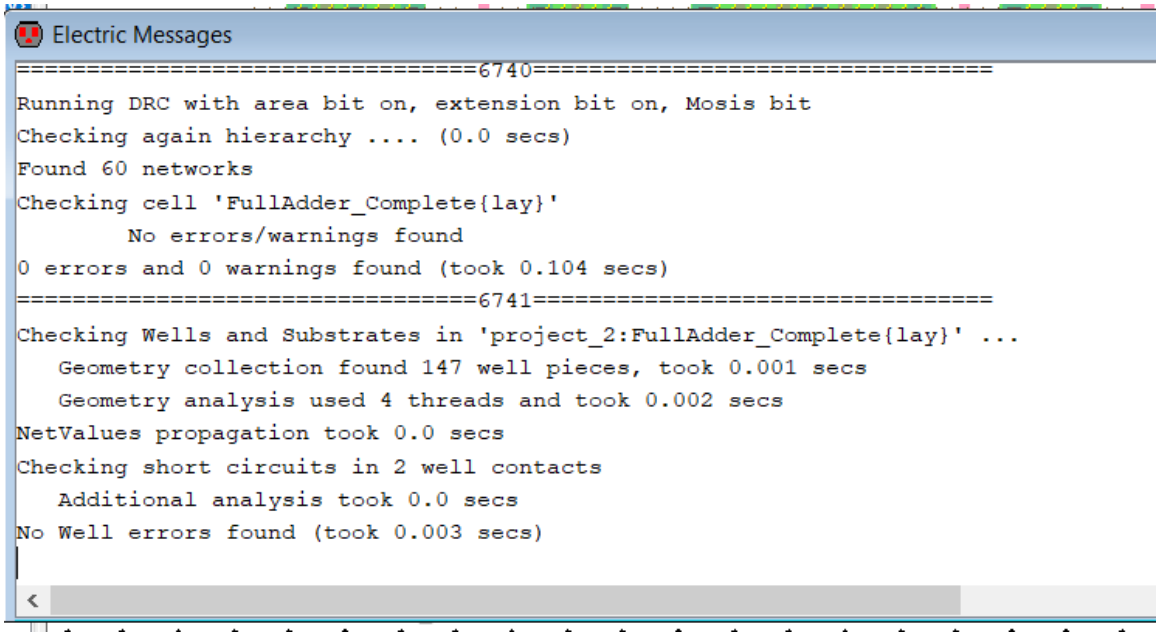

Figure 9: DRC and Well check for the Full Adder

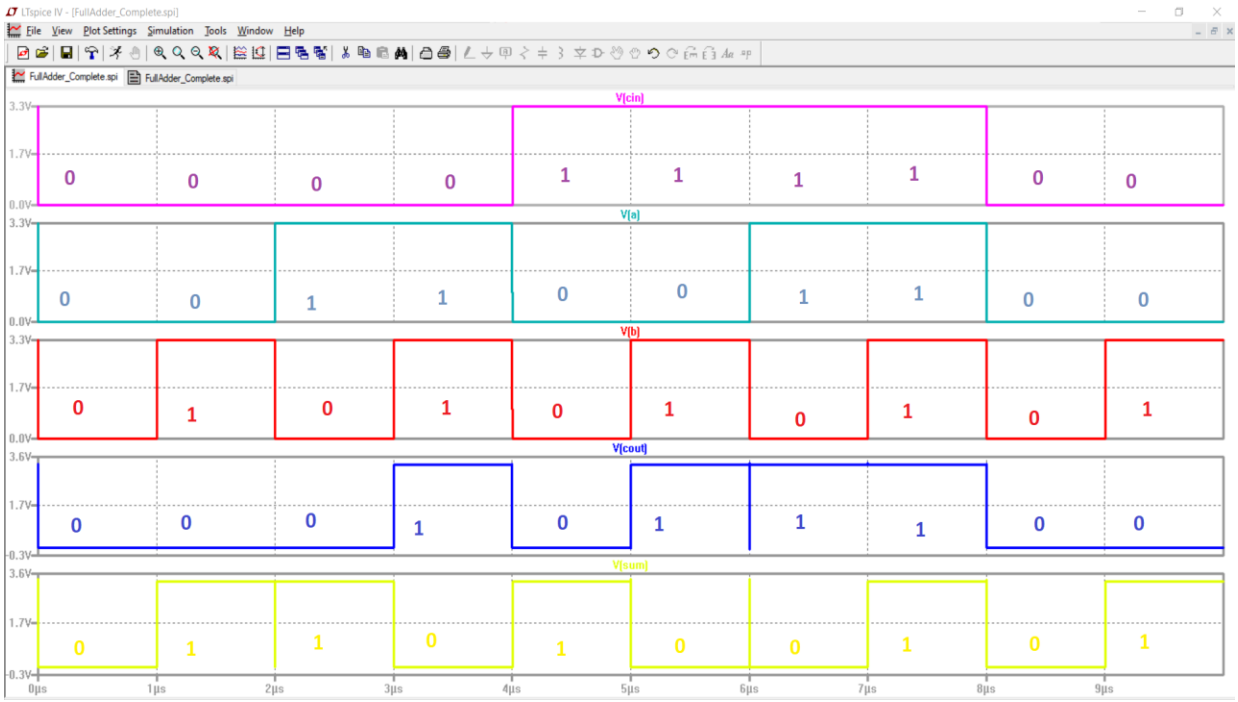

Figure 10: The LTSPICE simulation of the Full Adder

Upon observing the Ltspice simulation graph from Figure 10, we can conclude that the Full Adder design satisfy the Truth Table from the table 2. Thus we move on to build the Schematic of the 8-Bit Adder.

Section 3: Electric Circuit Schematic

Now that I Constructed the fundamental Full Adder, I cascaded 8 of the single Full Adders and combined them to construct the complete 8 Bit Ripple Carry Adder. Each of the Full Adder in the System will contain two inputs, A and B and a third input Cin and two output Sum and Carry out which carries the overflowed bit. Below is the figure of Schematic design of the 8-Bit Ripple Carry Adder. I have also included a snapshot of the DRC check for the schematic.

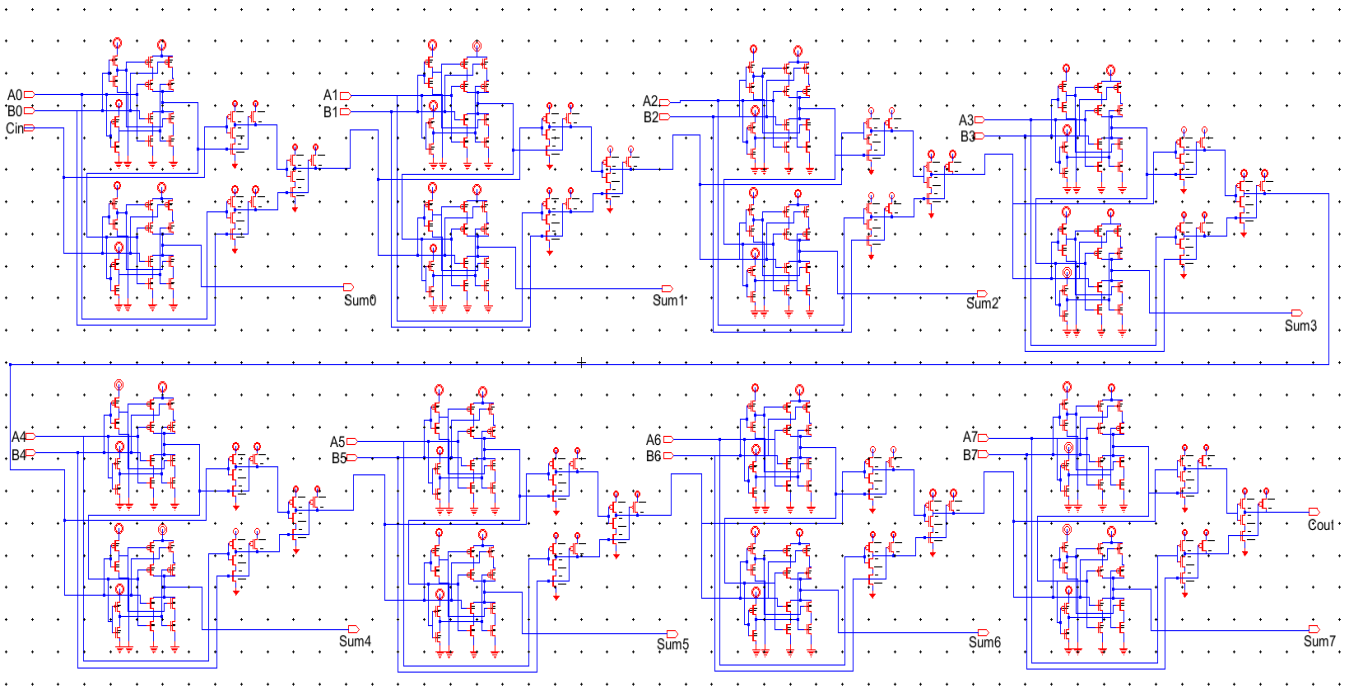

Figure 11: Schematic of 8-Bit Ripple Carry Adder.

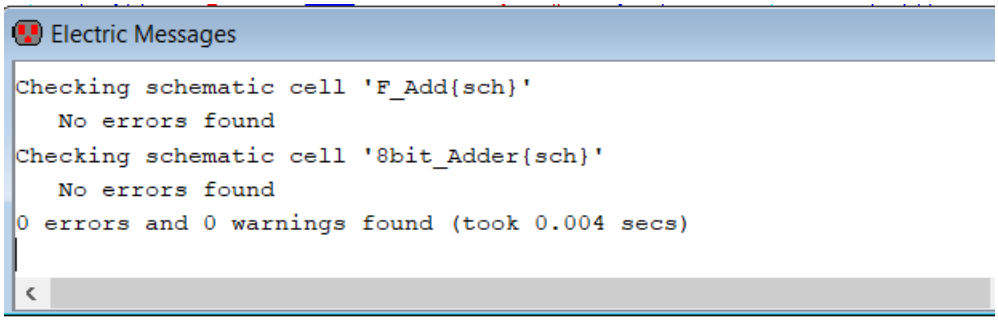

Figure 12: DRC check for 8-Bit Ripple Carry Adder

Section 4: LTSPICE simulations of Schematic:

After I completed the Schematic of the 8-Bit Ripple Carry Adder , I moved on to perform LTSPICE simulation to verify the verification Cases. We were instructed to provide the following verification cases (1) -32 + 107 = 75 , (2) 112 + 9 = 121 , (3) -28 – 16 = -44 and (4) -71 – 29 = -100 . We will use our spice code to initialize the inputs for all the test cases at once. The following table 4 shows the inputs in binary bits and expected outputs.

I have reorganized the test cases. I used the following LTSPICE code to initialize our Vdd to 3.3 , Gnd and 17 Inputs with input A0-A7 and Input B0-B7 also the input CIN. We are expecting 9 outputs, Output Sum0-Sum7 and Output Carry Out(COUT).

```
vdd vdd 0 dc 3.3
vgnd gnd 0 dc 0
Vcin Cin 0 dc 0
Va0 A0 0 pulse 3.3 0 0 0.01n 0.01n 60n 80n
Va1 A1 0 pulse 3.3 0 dc 0
Va2 A2 0 pulse 3.3 0 0 0.01n 0.01n 40n 60n
Va3 A3 0 pulse 3.3 0 0 0.01n 0.01n 60n 80n
Va4 A4 0 pulse 3.3 0 0 0.01n 0.01n 20n 40n
Va5 A5 0 dc 3.3
Va6 A6 0 pulse 3.3 0 60n 0.01n 0.01n 20n 20n
Va7 A7 0 pulse 3.3 0 20n 0.01n 0.01n 20n 60n
Vb0 B0 0 pulse 3.3 0 40n 0.01n 0.01n 20n 40n
Vb1 B1 0 pulse 3.3 0 20n 0.01n 0.01n 40n 80n
Vb2 B2 0 dc 0
Vb3 B3 0 pulse 3.3 0 40n 0.01n 0.01n 40n 40n
Vb4 B4 0 pulse 3.3 0 0 0.01n 0.01n 40n 60n
Vb5 B5 0 pulse 3.3 0 20n 0.01n 0.01n 20n 60n
Vb6 B6 0 pulse 3.3 0 20n 0.01n 0.01n 20n 60n
Vb7 B7 0 pulse 3.3 0 0 0.01n 0.01n 40n 80n
.tran 0 80n
.include C:\Electric\C5_models.txt
```

Figure 13:The SPICE code for our 8Bit Adder Schematic

Table 4: Initialization data Inputs and expected outputs for Verification cases.

	First Case	Second Case	Third Case	Fourth Case
Input: A A7-A0	-32 (11100000)	112 (01110000)	-28 (11100100)	-71 (10111001)
Input: B B7-B0	107 (01101011)	9 (00001001)	-16 (11110000)	-29 (11100011)
Output: Expected Sum S7-S0	75 (01001011)	121 (01111001)	-44 (11010100)	-100 (10011100)

Now the Snapshots of the generated waveforms from the LTSPICE simulation of our Schematic are provided below.

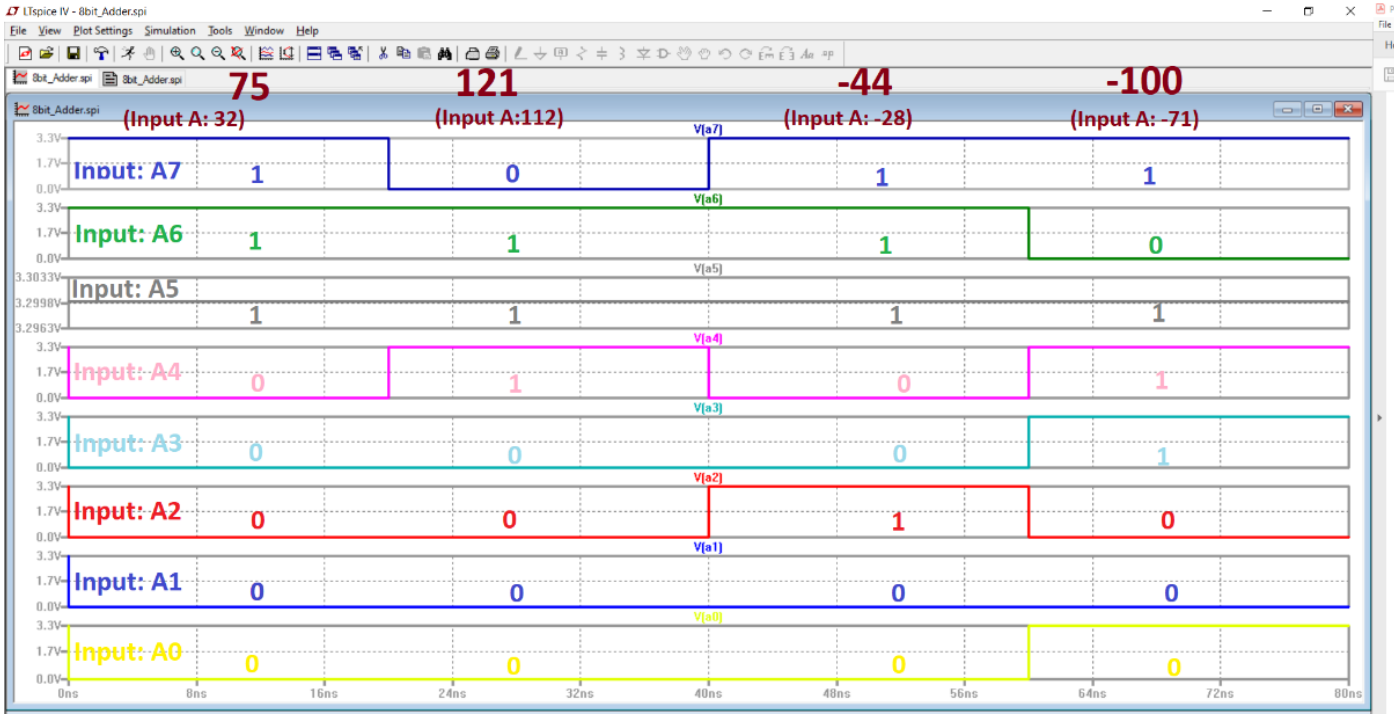

Figure 14: Input A0-A7 waveforms on LTSPICE for 8Bit Ripple Carry Adder Schematic

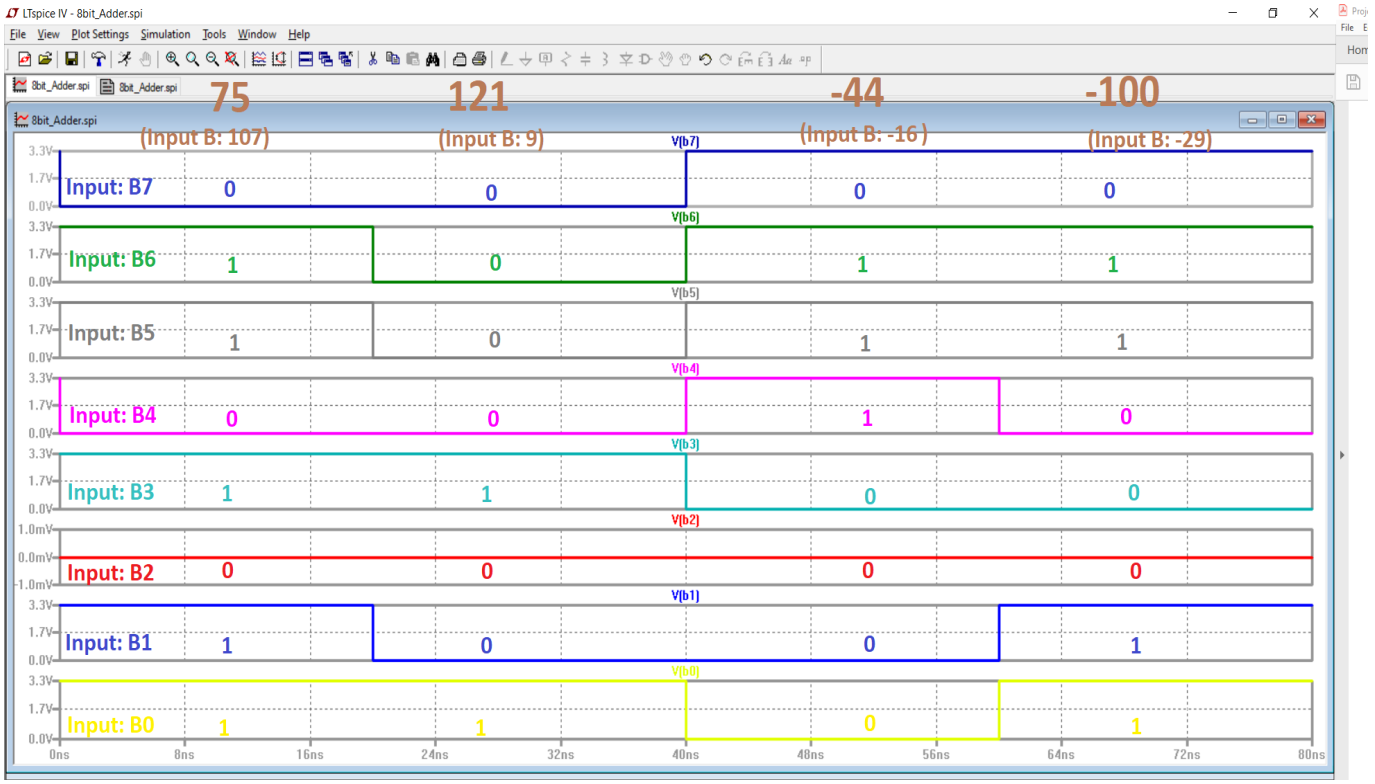

Figure 15:Input B0-B7 waveforms on LTSPICE for 8Bit Ripple Carry Adder Schematic

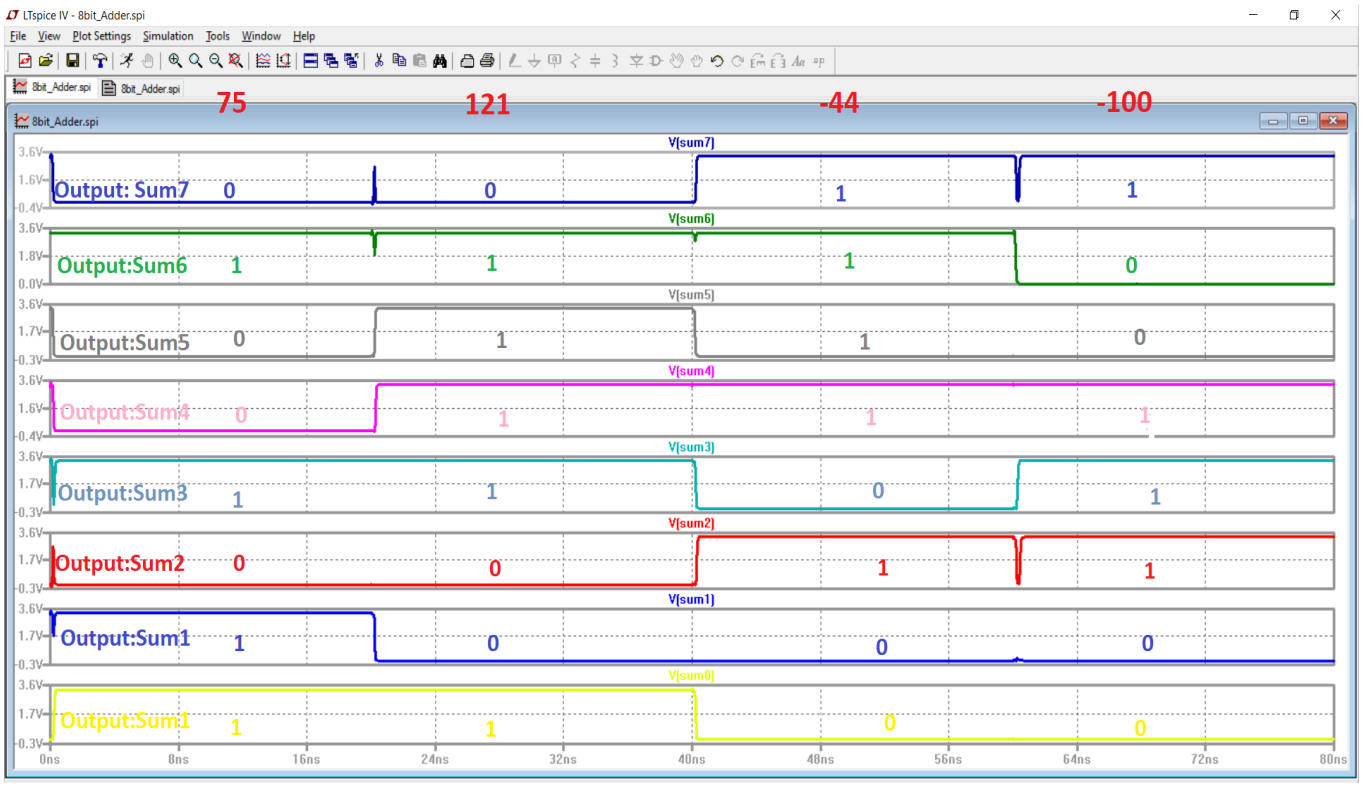

Figure 16: Outputs Sum7-Sum0 from LTSPICE waveform for 8-Bit Ripple Carry Adder Schematic

Table 5: Inputs and Outputs of the Verification Cases

	First Case	Second Case	Third Case	Fourth Case
Input: A A7-A0	-32 (11100000)	112 (01110000)	-28 (11100100)	-71 (10111001)
Input: B B7-B0	107 (01101011)	9 (00001001)	-16 (11110000)	-29 (11100011)
Output: Expected Sum	75 (01001011)	121 (01111001)	-44 (11010100)	-100 (10011100)

Output: LTSPICE SIM S7-S0	75 (01001011)	121 (01111001)	-44 (11010100)	-100 (10011100)

Section 5: IRSIM simulations of Schematic

After I completed the Schematic of the 8-Bit Ripple Carry Adder , I moved on to perform IRSIM simulation to verify the verification Cases. We were instructed to provide the following verification cases (1) -32 + 107 = 75 , (2) -28 – 16 = -44
(3) 112 + 9 = 121 and (4) -71 – 29 = -100 . We will use IRSIM to initialize the inputs for all the test cases at once. The following table 5 shows the inputs in binary bits and expected outputs.

I used the following IRSIM interface to initialize our Vdd to 3.3 , Gnd and 17 Inputs with input A0-A7 and Input B0-B7 also the input CIN. We are expecting 9 outputs, Output Sum0-Sum7 and Output Carry Out(COUT).

Table 6: IRSIM input and outputs of 8 BIT Ripple Carry Adder Schematic

	First Case	Second Case	Third Case	Fourth Case
Input: A A7-A0	107 (01101011)	-28 (11100100)	112 (01110000)	-71 (10111001)
Input: B B7-B0	-32 (11100000)	-16 (11110000)	9 (00001001)	-29 (11100011)
Output: Expected Sum	75 (01001011)	-44 (11010100)	121 (01111001)	-100 (10011100)

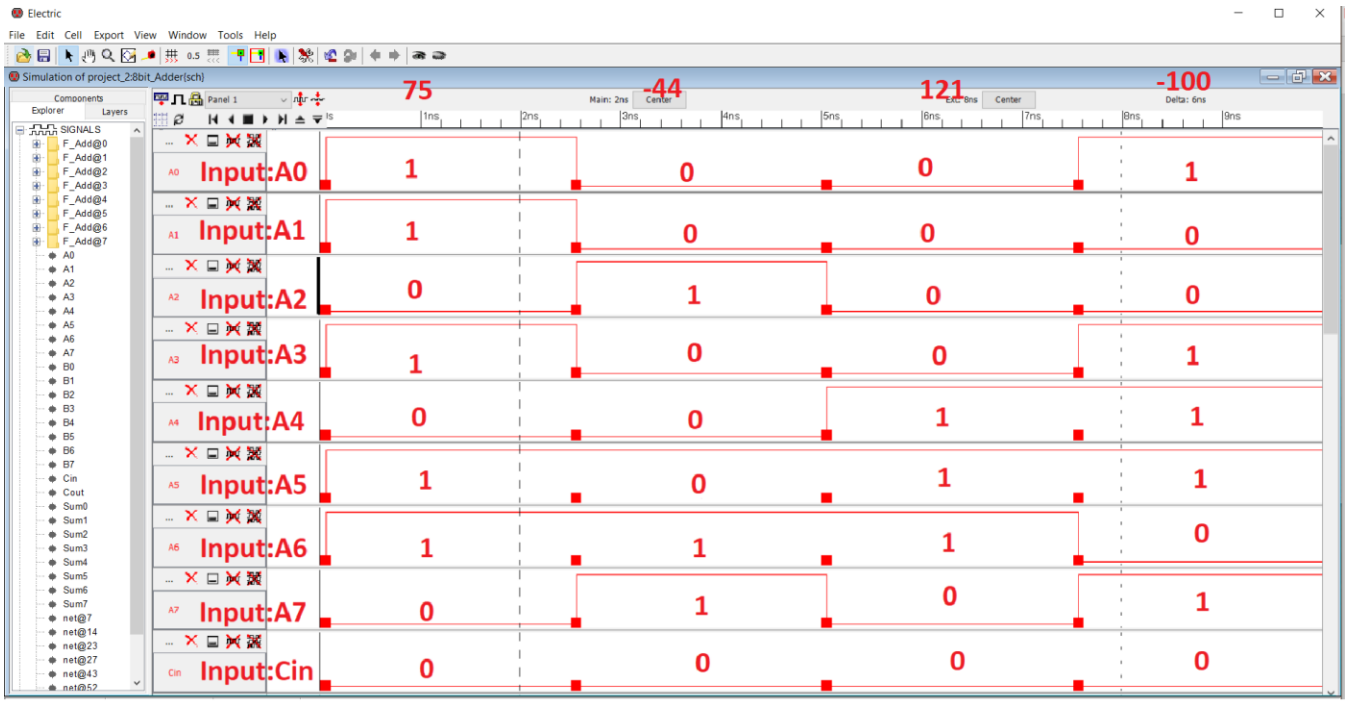

Figure 17: IRSIM Input A7-A0 waveforms of Schematic of 8-Bit Ripple Carry Adder

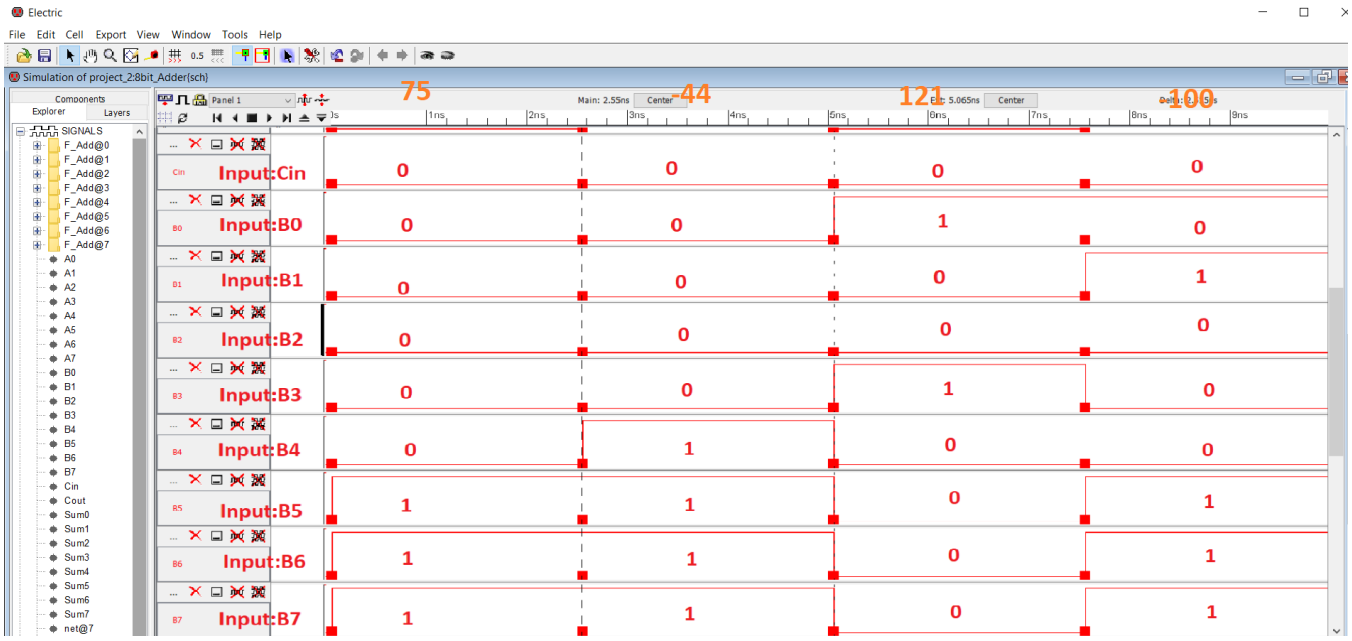

Figure 18: IRSIM Input B7-B0 and cin waveforms of Schematic of 8-Bit Ripple Carry Adder

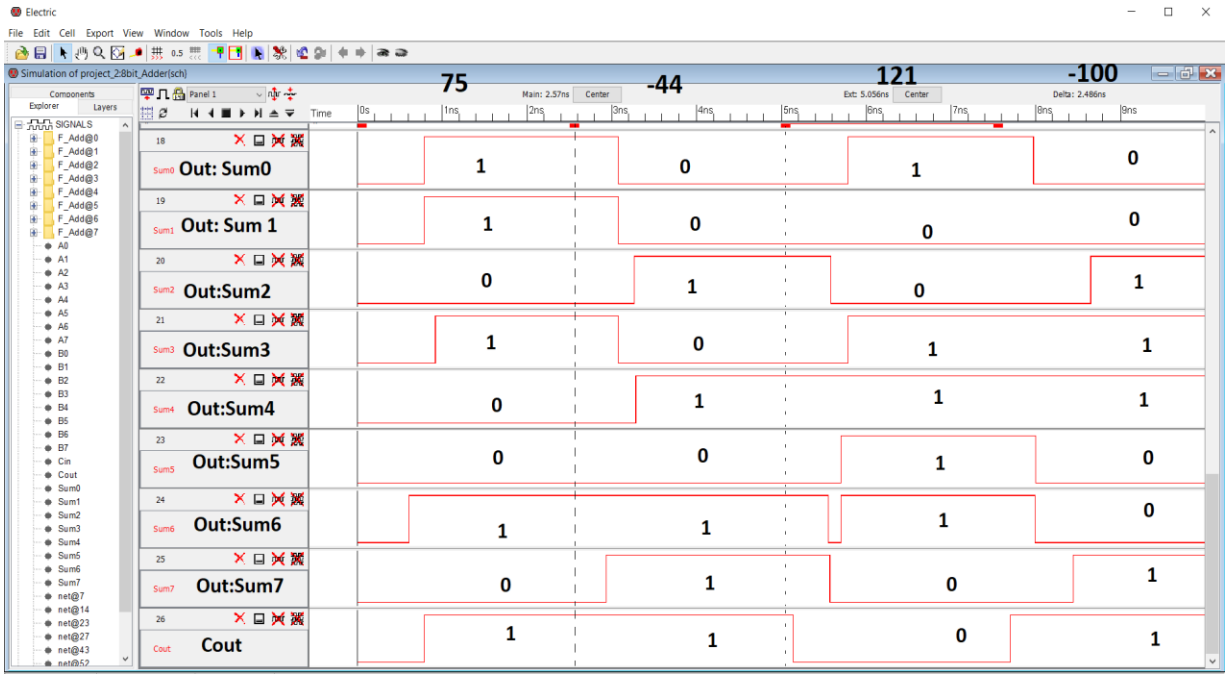

Figure 19:IRSIM Output Sum0-Sum7 and Cout waveforms of Schematic of 8-Bit Ripple Carry Adder

Table 7:Comparison of Outputs from IRSIM

	First Case	Second Case	Third Case	Fourth Case
Input: A A7-A0	107 (01101011)	-28 (11100100)	112 (01110000)	-71 (10111001)
Input: B B7-B0	-32 (11100000)	-16 (11110000)	9 (00001001)	-29 (11100011)
Output: Expected Sum	75 (01001011)	-44 (11010100)	121 (01111001)	-100 (10011100)
Output: IRSIM	75 (01001011)	-44 (11010100)	121 (01111001)	-100 (10011100)

Section 6: Electric Layout

Now that I Constructed the fundamental Full Adder, I cascaded 8 of the single Full Adders and combined them to construct the complete 8 Bit Ripple Carry Adder. Each of the Full Adder in the System will contain two inputs, A and B and a third input Cin and two output Sum and Carry out which carries the overflowed bit. Below is the figure of Layout design of the 8-Bit Ripple Carry Adder. I have also included a snapshot of the DRC and Well check for the Layout.

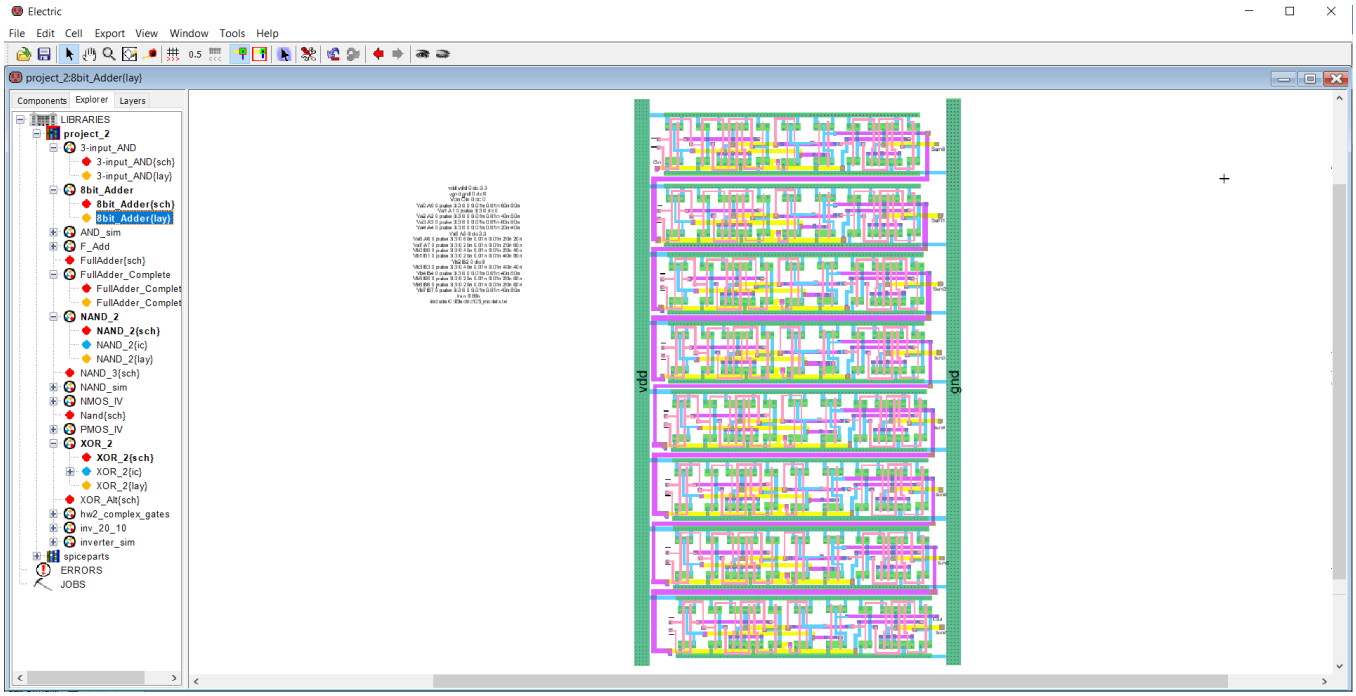

Figure 20: A overview of the Layout of 8-Bit Ripple Carry Adder

Above in Figure 20, the overview of the layout has been shown, Below in Figure 21, a closer overview of the Layout is presented.

Figure 22 provides a closer look at full adder 1-3 inside the Ripple Carry Adder, Figure 23 provides closer look at look at full adder 3-5 and Figure 24 does at full adder 5-7 and Figure 25 shows that the Layout design passes the DRC and Well checks in Electric.

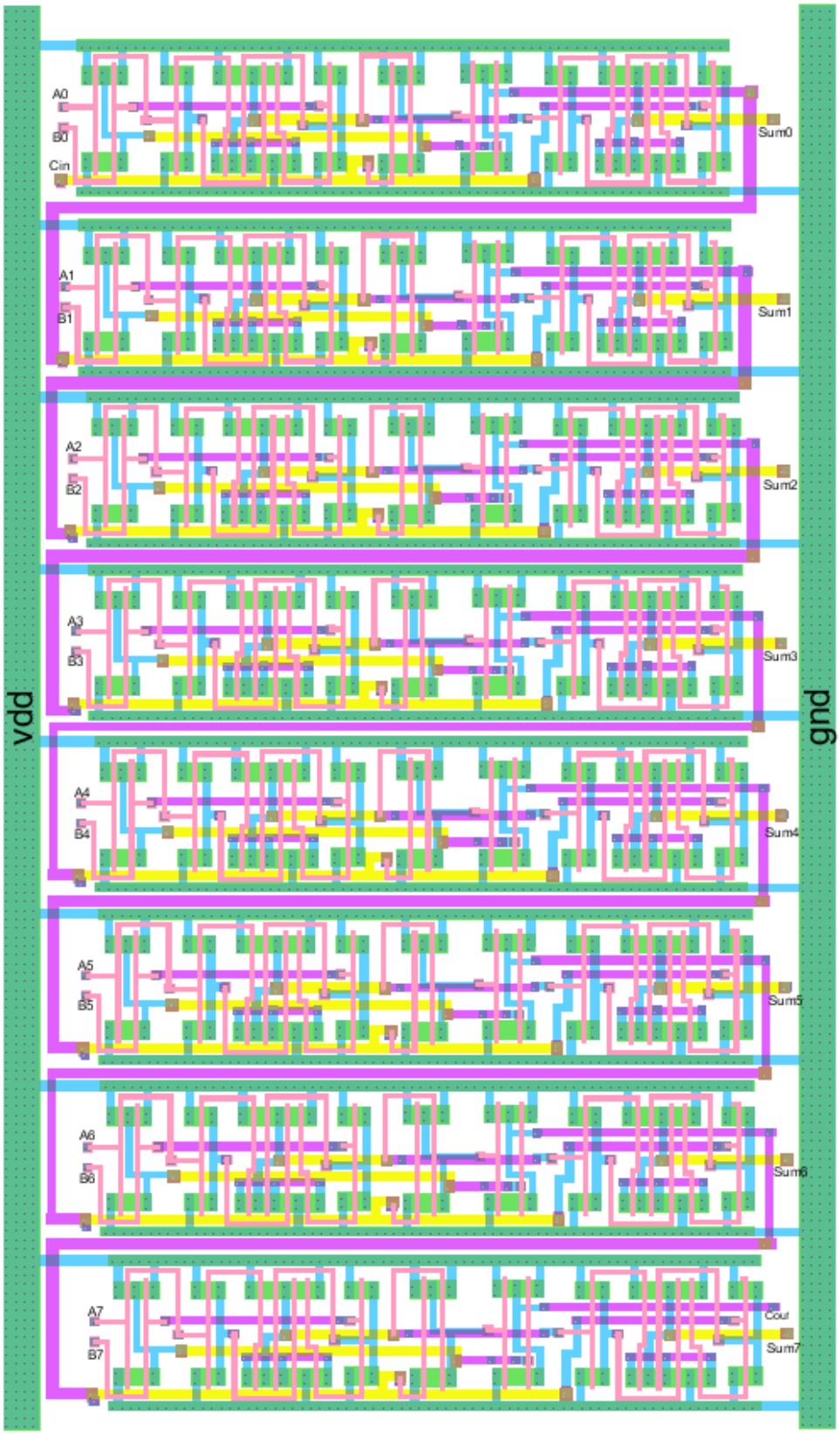

Figure 21: Overview of the Layout Design of the 8-Bit Ripple Carry Adder

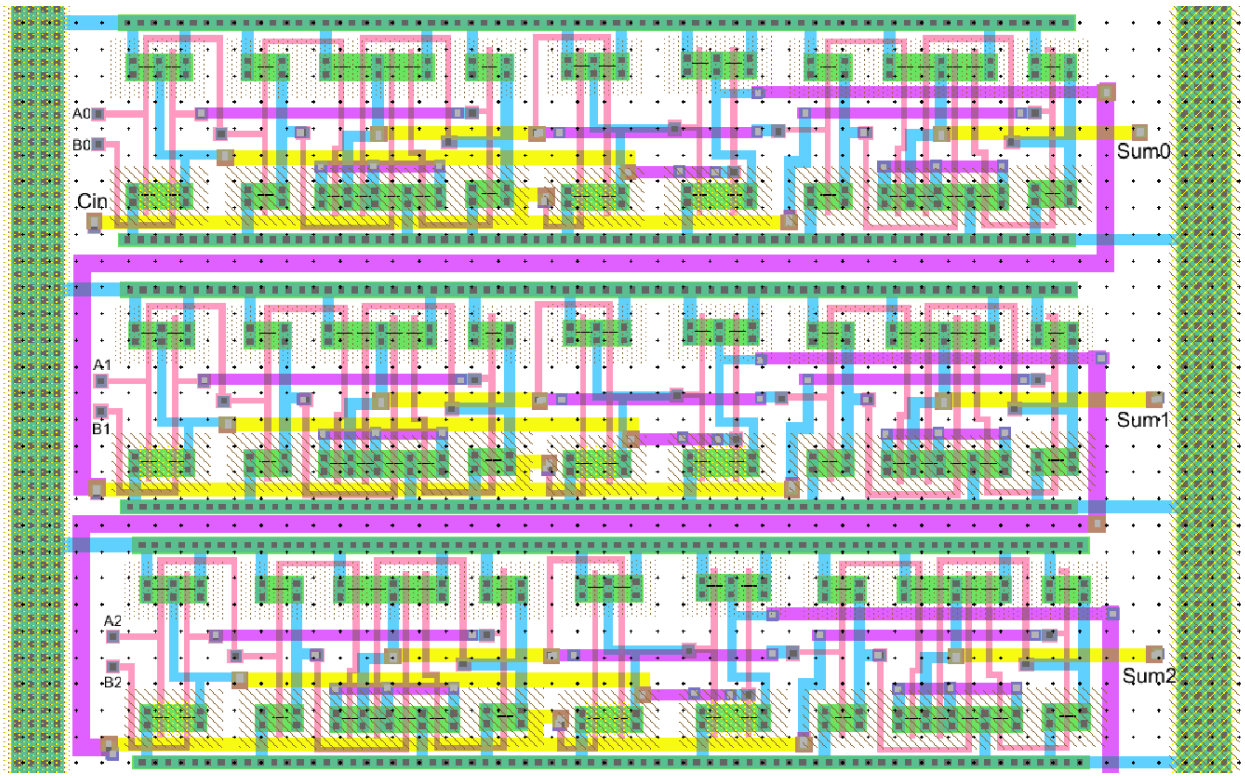

Figure 22:Zoomed in View for the Full Adder 1,2,3 inside the 8Bit RCA

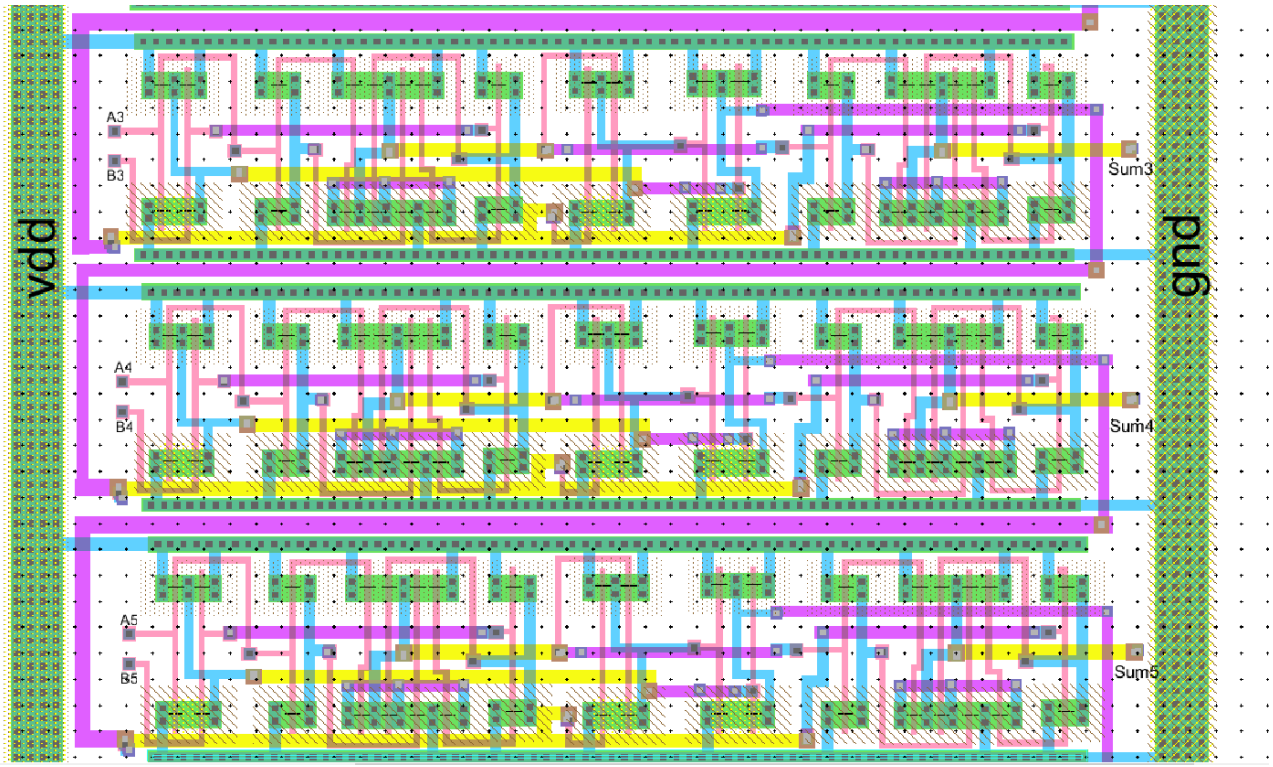

Figure 23:Zoomed in View for the Full Adder 3,4,5 inside the 8Bit RCA

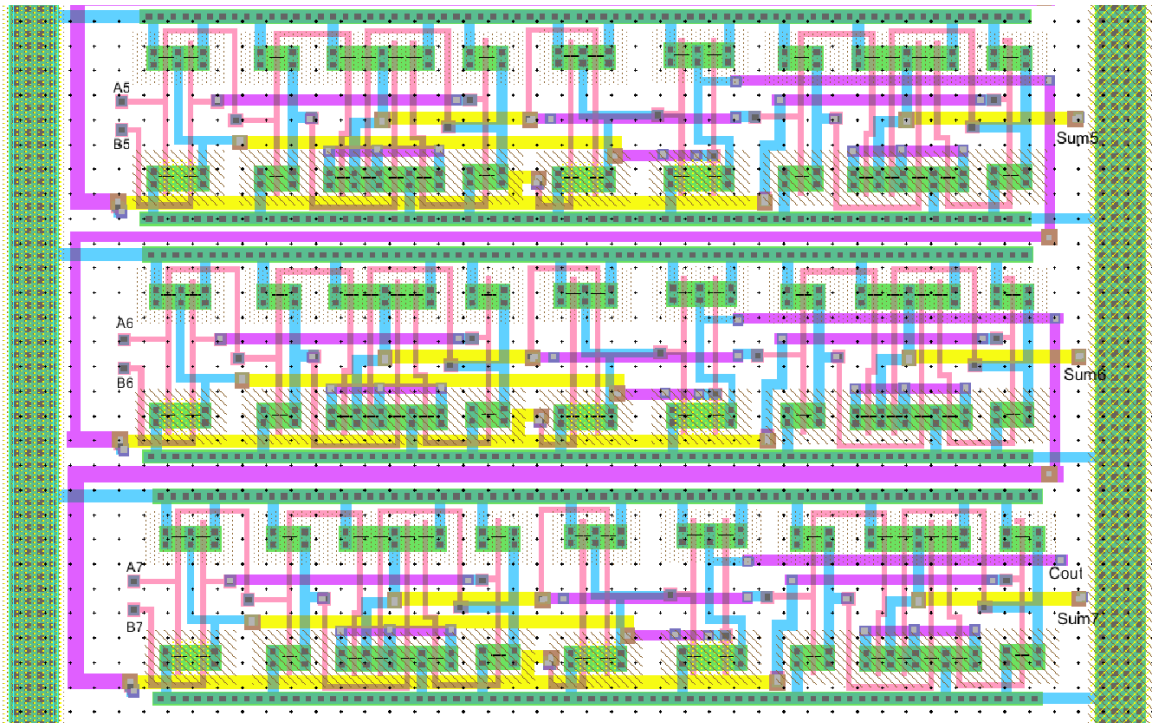

Figure 24:Zoomed in View for the Full Adder 5, 6, 7 inside the 8Bit RCA

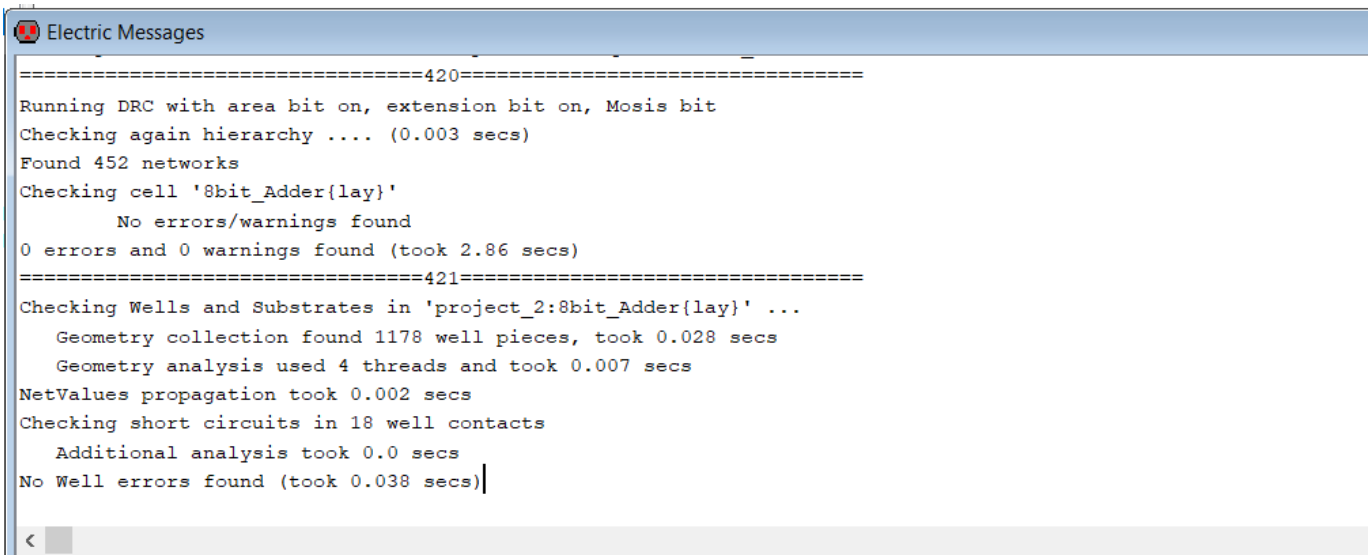

Figure 25: DRC and Well Check for 8Bit Ripple Carry Adder Layout

Section 7: LTSPICE for Layout

After I completed the Layout of the 8-Bit Ripple Carry Adder , I moved on to perform LTSPICE simulation to verify the verification Cases. We were instructed to provide the following verification cases (1) -32 + 107 = 75 , (2) 112 + 9 = 121 , (3) -28 – 16 = -44 and (4) -71 – 29 = -100 . We will use our spice code to initialize the inputs for all the test cases at once. The following table 9 shows the inputs in binary bits and expected outputs.

I have reorganized the test cases. I used the following LTSPICE code to initialize our Vdd to 3.3 , Gnd and 17 Inputs with input A0-A7 and Input B0-B7 also the input CIN. We are expecting 9 outputs, Output Sum0-Sum7 and Output Carry Out(COUT). The Transient analysis will continue for about 80ns.

```
vdd vdd 0 dc 3.3
vgnd gnd 0 dc 0
Vcin Cin 0 dc 0
Va0 A0 0 pulse 3.3 0 0 0.01n 0.01n 60n 80n
Va1 A1 0 pulse 3.3 0 dc 0
Va2 A2 0 pulse 3.3 0 0 0.01n 0.01n 40n 60n
Va3 A3 0 pulse 3.3 0 0 0.01n 0.01n 60n 80n
Va4 A4 0 pulse 3.3 0 0 0.01n 0.01n 20n 40n
Va5 A5 0 dc 3.3
Va6 A6 0 pulse 3.3 0 60n 0.01n 0.01n 20n 20n
Va7 A7 0 pulse 3.3 0 20n 0.01n 0.01n 20n 60n
Vb0 B0 0 pulse 3.3 0 40n 0.01n 0.01n 20n 40n
Vb1 B1 0 pulse 3.3 0 20n 0.01n 0.01n 40n 80n
Vb2 B2 0 dc 0
Vb3 B3 0 pulse 3.3 0 40n 0.01n 0.01n 40n 40n
Vb4 B4 0 pulse 3.3 0 0 0.01n 0.01n 40n 60n
Vb5 B5 0 pulse 3.3 0 20n 0.01n 0.01n 20n 60n
Vb6 B6 0 pulse 3.3 0 20n 0.01n 0.01n 20n 60n
Vb7 B7 0 pulse 3.3 0 0 0.01n 0.01n 40n 80n
.tran 0 80n
.include C:\Electric\C5_models.txt
```

Figure 26: SPICE code for the Layout Simulations.

Table 8:Initialization data Inputs and expected outputs for Verification cases in LTSPICE layout

	First Case	Second Case	Third Case	Fourth Case
Input: A A7-A0	-32 (11100000)	112 (01110000)	-28 (11100100)	-71 (10111001)
Input: B B7-B0	107 (01101011)	9 (00001001)	-16 (11110000)	-29 (11100011)
Output: Expected Sum S7-S0	75 (01001011)	121 (01111001)	-44 (11010100)	-100 (10011100)

We used the SPICE code in figure 26 in such a way that it accommodate the input data shown in Table 9 above. After we ran the simulation following waveforms were produced.

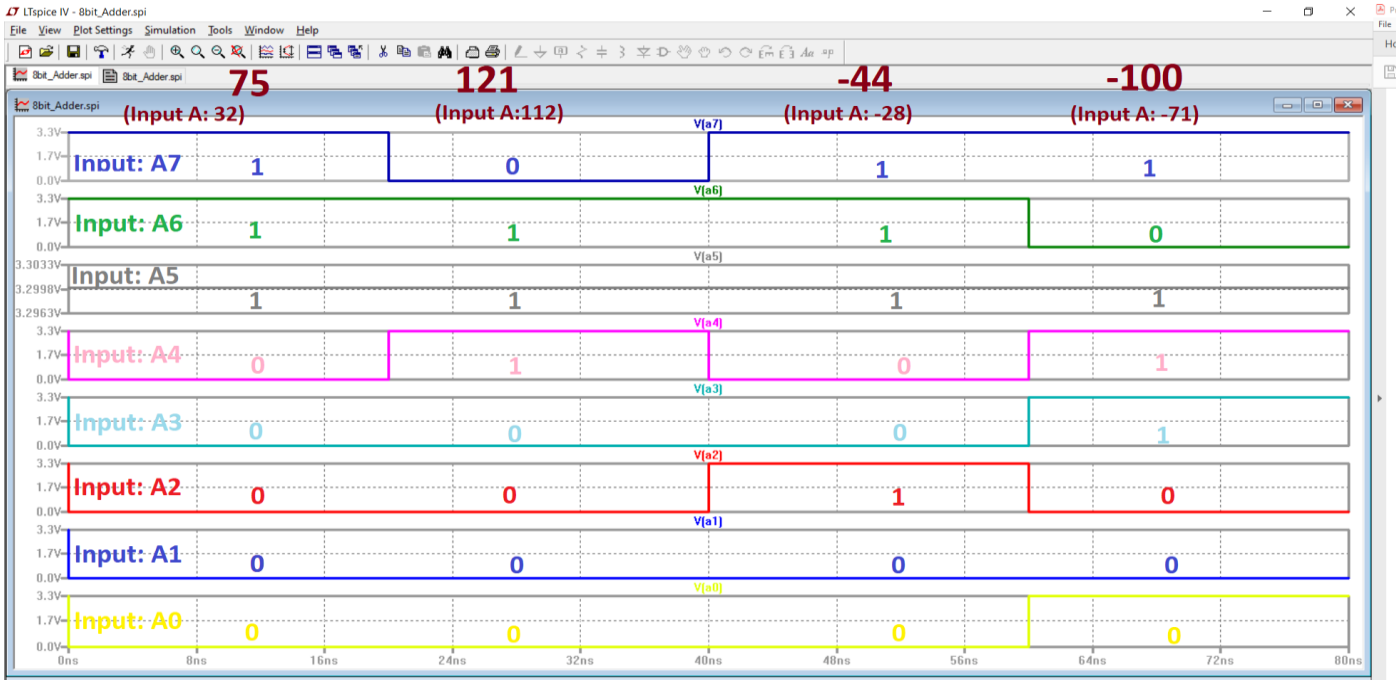

Figure 27: LTSPICE waveforms of LAYOUT of 8-Bit Ripple Carry Adder(Inputs A7-A0)

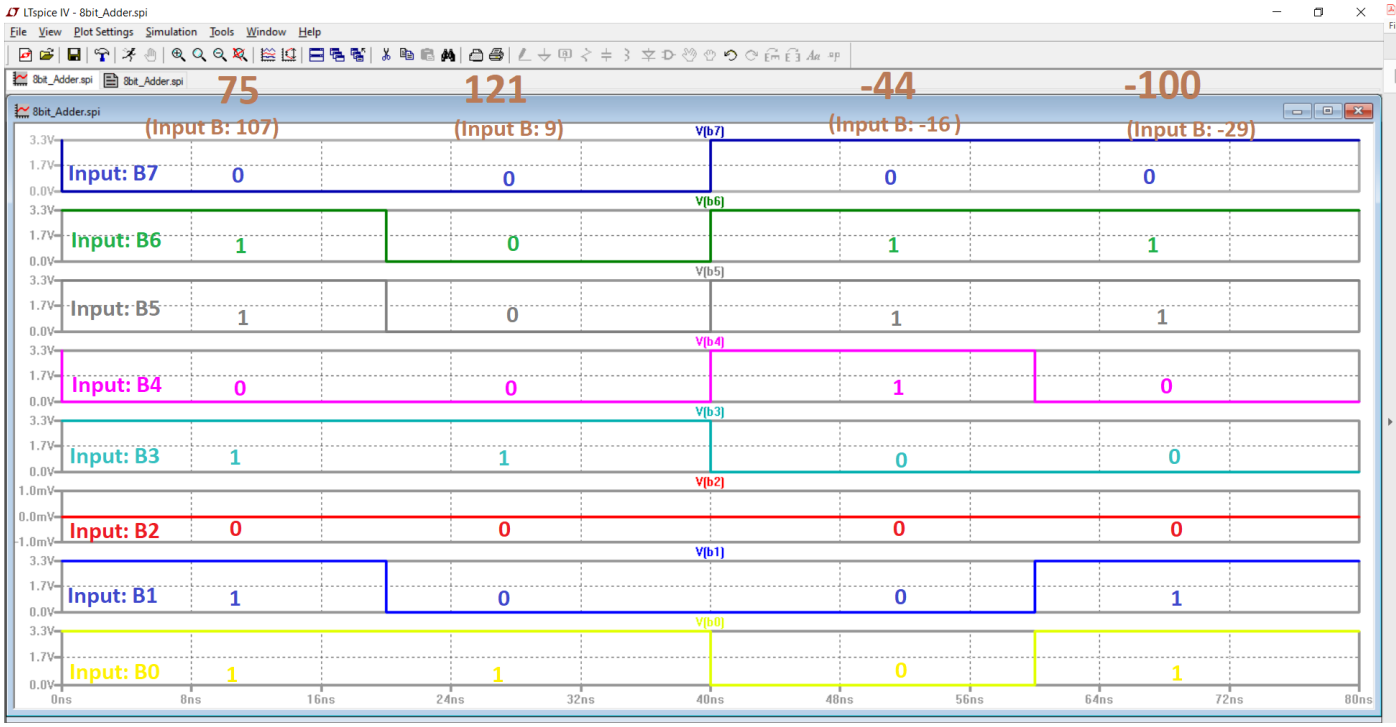

Figure 28:LTSPICE waveforms of LAYOUT of 8-Bit Ripple Carry Adder(Inputs B7-B0)

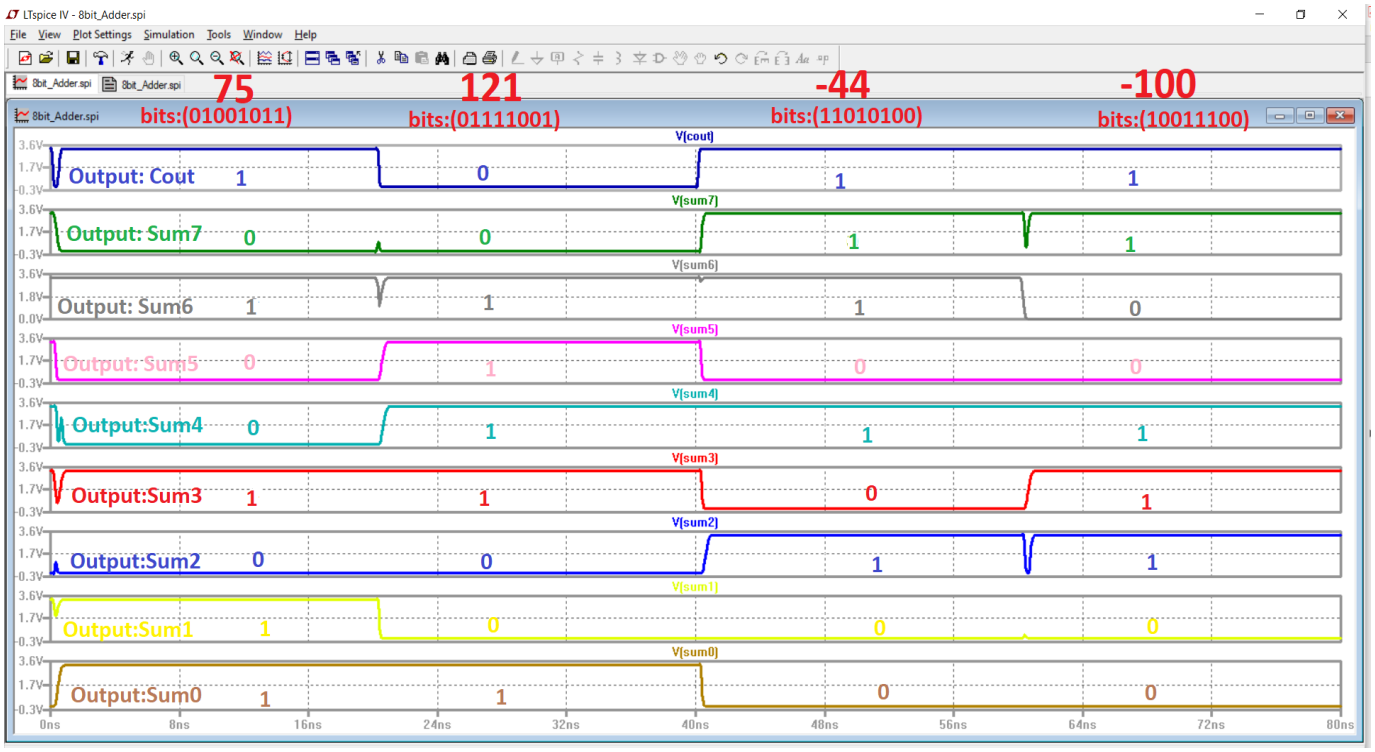

Figure 29:LTSPICE Output Sum0-Sum7 Waveforms for Layout of the 8Bit Ripple Carry Adder

Table 9: Verification cases and Output for LTSPICE of 8bit RCA Layout

	First Case	Second Case	Third Case	Fourth Case
Input: A A7-A0	-32 (11100000)	112 (01110000)	-28 (11100100)	-71 (10111001)
Input: B B7-B0	107 (01101011)	9 (00001001)	-16 (11110000)	-29 (11100011)
Output: Expected Sum	75 (01001011)	121 (01111001)	-44 (11010100)	-100 (10011100)

Output: LTSPICE Sim S7-S0	75 (01001011)	121 (01111001)	-44 (11010100)	-100 (10011100)

Upon observing the table 9 and the outputs from figure 29, we can conclude that the LTSPICE Simulations for the Layout of 8 Bit Ripple carry Adder satisfies all four verification.

Section 8: IRSIM Simulation of Layout

After I completed the Layout of the 8-Bit Ripple Carry Adder , I moved on to perform IRsim simulation to verify the verification Cases. We were instructed to provide the following verification cases (1) -32 + 107 = 75 , (2) -28 – 16 = -44 (3) 112 + 9 = 121 and (4) -71 – 29 = -100 . We will use IRSIM to initialize the inputs for all the test cases at once. The following table 5 shows the inputs in binary bits and expected outputs.

I used the following IRSIM interface to initialize our Vdd to 3.3 , Gnd and 17 Inputs with input A0-A7 and Input B0-B7 also the input CIN. We are expecting 9 outputs, Output Sum0-Sum7 and Output Carry Out(COUT).

Table 10: IRSIM Input and Expected Output For 8Bit Ripple Carry Adder Layout

	First Case	Second Case	Third Case	Fourth Case
Input: A A7-A0	107 (01101011)	-28 (11100100)	112 (01110000)	-71 (10111001)
Input: B B7-B0	-32 (11100000)	-16 (11110000)	9 (00001001)	-29 (11100011)
Output: Expected Sum	75 (01001011)	-44 (11010100)	121 (01111001)	-100 (10011100)

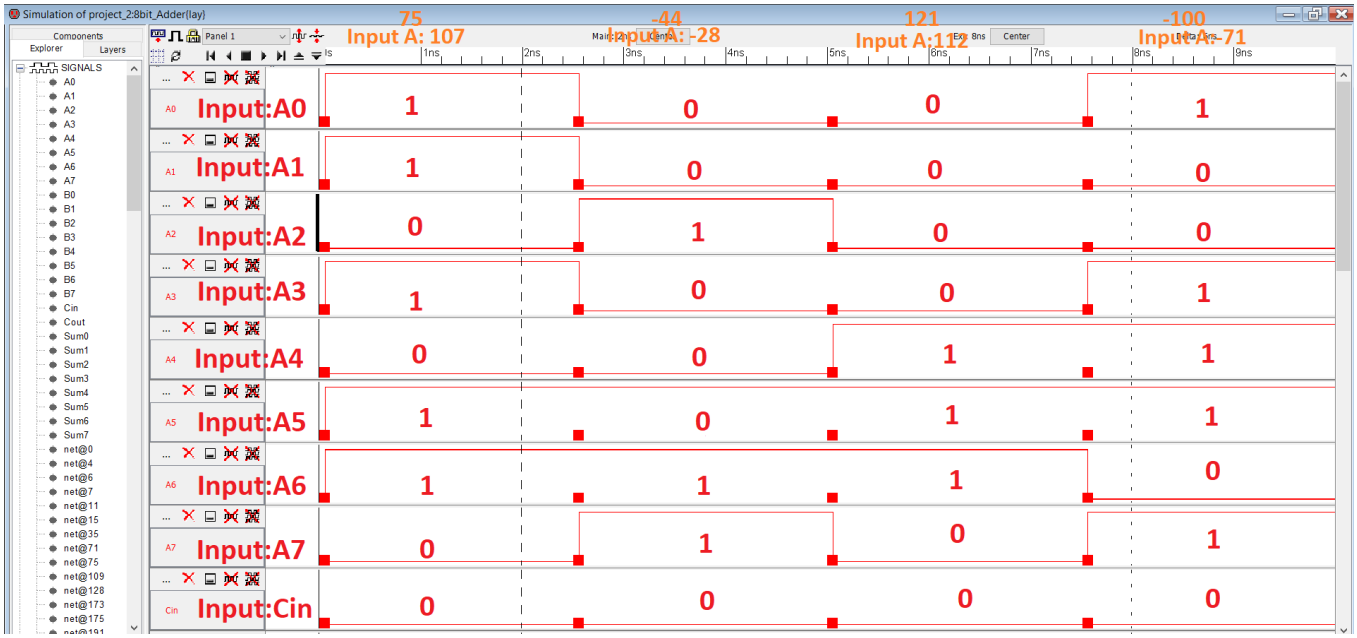

Figure 30: Input A0-A7 and Cin for IRSIM of the 8Bit Ripple carry Adder Layout

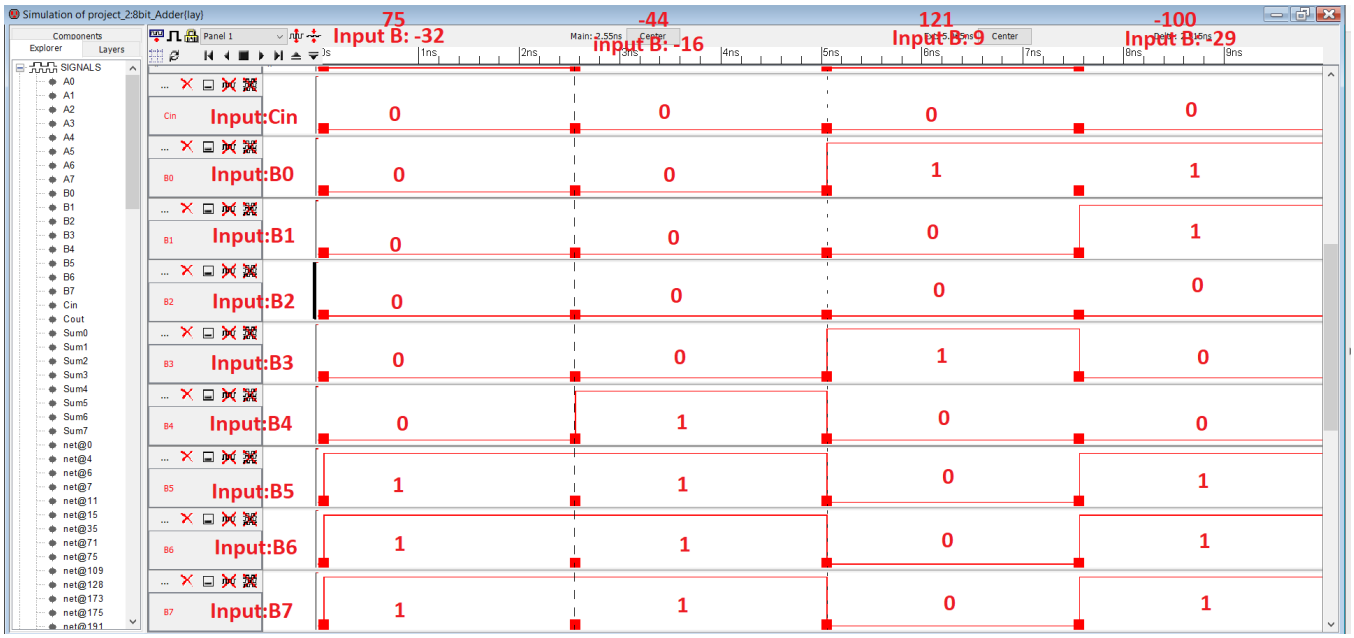

Figure 31:Input B0-B7 for IRSIM of the 8Bit Ripple carry Adder Layout

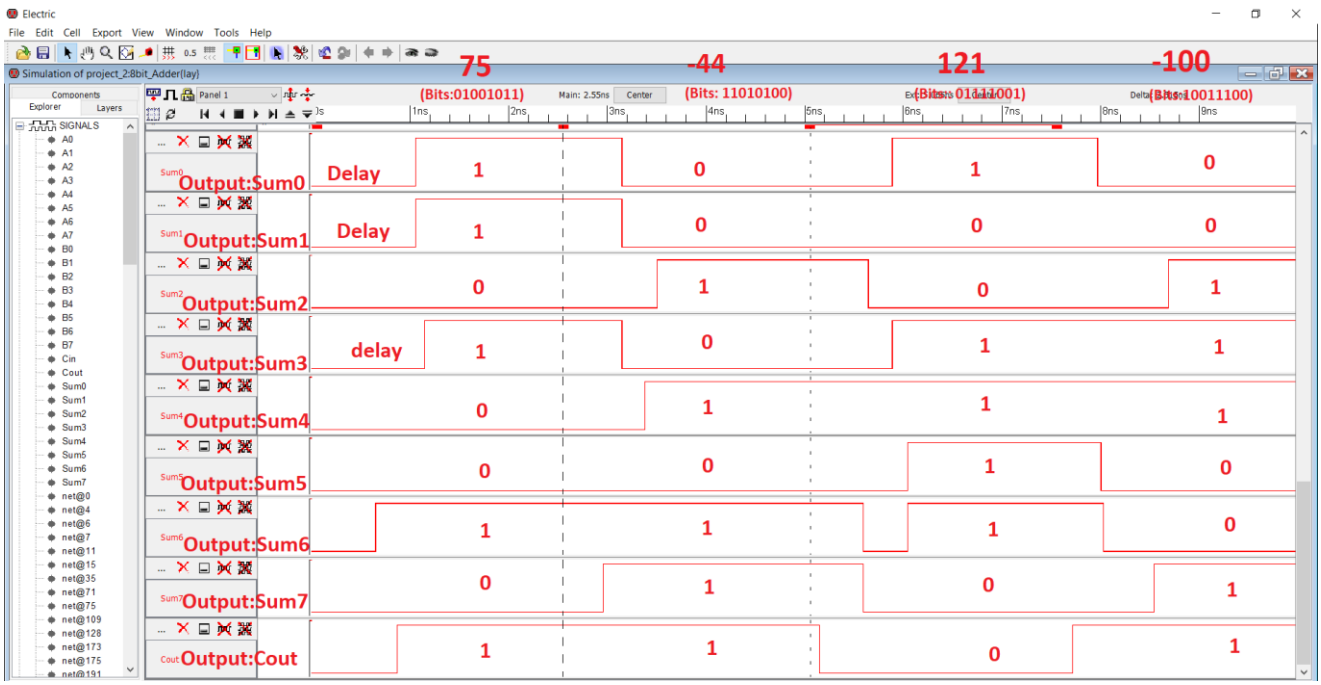

Figure 32:OUTPUT Sum0-Sum7 and Cout for IRSIM of the 8Bit Ripple carry Adder Layout

Upon Observing the Outputs from Figure 32, I can conclude that it matches exactly to the expected output for our verification cases stated in the table 10. Thus the Layout passes all the four verification cases.

Section 9: Summary of Measurements

Table 11: Summary of Measurement Rise time, Fall Time and Propagation delay

	Schematic LTSPICE	Layout LTSPICE	Schematic IRSIM	Layout IRSIM
Rise Time	0.2 ns to 0.4 ns	0.4 ns to 0.8 ns		
Fall Time	0.17 ns to 0.35ns	0.3 ns to 0.45ns		
Propagation Delay	0.18 ns to 0.375 ns	0.35 ns to 0.625 ns	0.25 ns to 3.196 ns	0.30 ns to 3.5ns

From the table 11 above we can surmise that the Schematic is faster in LTSPICE in respect to the Layout design. If we looks at the delay times the delay in the Schematics IRSIM simulations are by far less than the delays found from the simulations of the Layout design. Same case for the LTSPICE, there is notable less delay in the Schematic Simulations than in the Layout.

Table 12: Measurements of Transistor Sizes, Power Dissipation and Total chip area

	Schematic	**Layout**
Transistor Sizes (W/L)	PMOS(10/2),NMOS(10/2)	PMOS(10/2), NMOS(10/2)
Power Dissipation	3.3v* 21 μA = .0007 watt	3.3v * 40 μA = 0.000132 Watt
Total Chip Area		117821.65625 μm^2

The table 12 show the Transistor sizes of PMOS being 10/2 and NMOS of 10/2 consistent over both my schematic and layout design. The total Chip are was measured in the electric as 473.5x812.5 and our lambda was 175.0 nm. thus width of 82862.5 nano meters and length of 142187.5 nanometers. Which get converted to width of 82.86 micrometers and length of 142.1875 micrometers. So the total chip area in the Layout becomes (82.86*142.1875) = 117821.65625 μm^2

Section 10: Comparison of Schematic & Layout:

In case of LTSPICE, by comparing Electric Schematic in Figure 21 with Electric Layout in Figure 21, the way the input and output reacted given the Spice Code seems to be the very similar. At the same time the inputs and outputs of the verification cases from from Table 4 of Electric Schematic and Table 8 of Electric Layout are the same too. As it passes the verification cases then the design also satisfy the project requirements. The only difference between the two figures would be the rise time, fall time, and propagation delay discussed in the Summary measurement section from Table 11.

In case of IRSIM, we start by comparing Figure 14,15 and 16 which are the IRSIM simulations of the electric Schematic shown in Figure 11, with Figure 17,18,19 which are the IRSIM simulation of the Electric Layout, we can observe, that the Input and Output Logics appears to be same evident with our verification inputs. Both table 9 and Table 7 containing the IRSIM findings confirms this and verifies our Ripple carry adder design. Again the only notable difference between the figures is the propagation delay, which is discussed in the measurement section and seen in the table 11.

Thus I can determine that the logic of both layout and schematic design for my 8-Bit Ripple Carry Adder is being verified by both LTSPICE and IRSIM however there is at time notable difference between them as evident from our findings from the Rise and Fall Times and Propagation delay both of which presented in the measurement sections.

Section 11: Conclusion

For the, I have designed a CMOS of 8-Bit Ripple Carry Adder by using Electric. I cascaded 8 Full Adder to construct the 8-Bit Ripple Carry Adder. But First I built a Full Adder which contains 2 XOR gates and 3 NAND gates. I constructed both Schematic and Layout. After the design I generated waveform using both LTSPICE and IRSIM to verify the Verification Cases. We tested all 4 verification cases and both of my designs satisfied the test properties.

Then I moved on to compare the waveforms I gathered from both of my designs. Generating waveforms from two design using two different tools allowed me to analyze and observe their differences. As mentioned in the comparison section 9, time the inputs and outputs of the verification cases from Table 4 of Electric Schematic and Table 8 of Electric Layout are the same too as it passes the verification cases where the only difference between the findings from two simulations would be the rise time, fall time, and propagation delay. I determined that the Schematic is overall more faster and has less delay due to uniform voltage distribution and also due to the fact that the schematic designs is more plain and straight forward where Layout involve handling more intricate part which increases possibility of inefficient design.

References:

I. Design, Layout and Simulation of NAND/XOR/Full Adder by Baker

http://cmosedu.com/jbaker/courses/

II. Adder (Electronics) [Online]:
https://en.wikipedia.org/wiki/Adder_(electronics)

III. Full Adder source

https://www.sciencedirect.com/topics/computer-science/full-adder

IV. Ripple Carry and Carry Look Ahead Adders

https://www.ece.uvic.ca/~fayez/courses/ceng465/lab_465/project1/adders.pdf