EE457: Digital IC Design
Spring Semester 2019
Project 3 Report Cover Sheet

Due 5/13/2019

PROJECT TITLE: 32 Bit Adder/Subtractor Using Look Ahead Carry Logic

PROJECT Number: 4/A

Student Names: Sebastian Grygorczuk

Check for Completion	Topics	Grades
X	Section 1: Executive Summary	/5
X	Section 2: Introduction and Background	/5
X	Section 3: Electric Circuit Schematic	/10
X	Section 4: Detailed Electric Layout	/25
X	Section 5: IRSIM Logic Simulations and Measurements for Layout and Schematic (must provide comparisons between the two)	/10
X	Section 6: LTSPICE code and parasitic extractions with Analysis and measurements	/15
X	Section 7: Measurements in LTSPICE for delays for Layout and Schematic (must provide comparisons between the two)	/15
X	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.)	Power /2 Delay /2 Area /2 #Trans /4
X	Section 9: Conclusion and References	/5
	Penalty	
	TOTAL	/100

Section 1: Executive Summary

I created all the necessary components and was able to fully connect the schematic version however I wasn't able to finish the 32 Bit CLA layout as the AND and OR gates I've constructed lost their characteristics with a large number of inputs due to parasitics. I've made roughly 100 components in both schematic and layout, creating 32 different AND and OR gates and I used those to create 31 individual carry signals, along with that I've created an Adder/Subtractor which is made up of three XOR gates and an AND gate. I performed a deep analysis of the components going over the AND, OR, Carry functions and Adder/Subtractor and found that my AND and Or gate builds created a giant net of capacitors. The OR gates became very sensitive and the smallest amount of voltage from any of the inputs would turn them on and keep them high for prolonged periods of time meanwhile the AND gates required a significant amount of time to charge up before turning on. I didn't believe this would occur as while creating the units I've tested the smaller versions and they behaved appropriately and I believed that the larger version would keep the same characteristics. I was wrong to assume that and should have tested all of the components while creating them. This lead to the carry function inheriting the flaws of the gates. I know it's not the design of the system as a whole as the schematic version works correctly nevertheless the layout is flawed due to parasitics. The discovery of the parasitics is deeper explored in sections 6 and 7. To correct this mistake I would have to reconstruct all of the AND and OR gates for the layout to be able to complete the CLA, this task would add another 40-50 hours to redo which I just did not have the time to do as I've put in around 60-70 hours into the project already.

Section 2: Introduction and Background

In this project I tackled the issue of creating a 32 Bit Adder/Subtractor using Carry Look Ahead (CLA) Logic. The difference between a 32 Bit Adder/Subtractor using CLA Logic and a Adder/Subtractor using Ripple Carry is that the CLA Logic that Ripple Carry uses the output carry of the previous adder to calculate its sum meaning that the very last sum has to wait for all of the sums before it to be calculated. This means a lot of time being wasted CLA Logic makes it so that all of the Carry signals are calculated at the same time and then the sum are also calculated at the same time. The negative to this is that CLA Logic more complicated and requires many more components to make it possible growing larger with every bit that need to be calculated.

The way the CLA Logic works is by having the inputs A XOR B give out a Propagation signal and A AND B give out the Generated Propagation. These signals are the connected in series of

complex functions to calculate carry of each bit individually. Image below is shows the idea behind CLA with up to 4 Bits

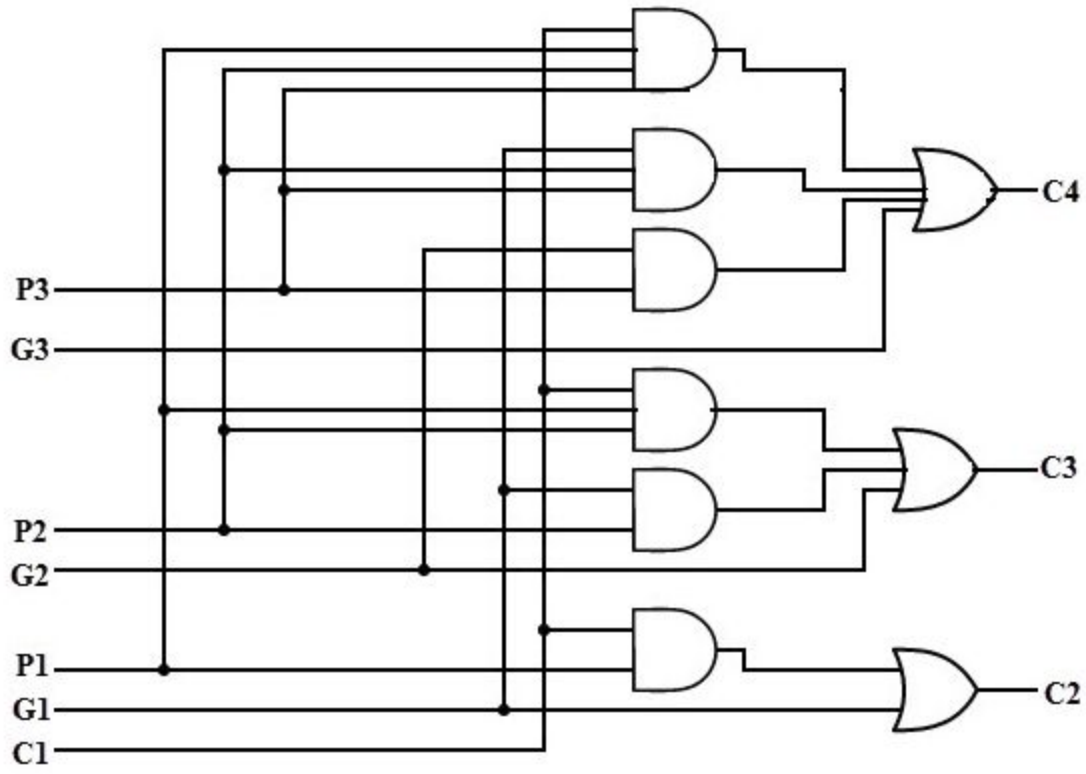


Figure 1. CLA from The Electronic Hub

The formula behind this logic go as such

$P_i = A_i$ XOR B_i
$S_i = P_i$ XOR C_i
$G_i = A_i$ AND B_i
$C_{i+1} = G_i$ OR P_i AND C_i

It's a recursive function where the final Carry is a conjoined with all the carries before it and every Carry before that has its own function that works in the same fashion. As One can see this gets complicated very fast as we look at the expansion of the function for the first four bits.

C1 = G0 + P0*C0

C2 = G1 + P1*C1 = G1 + P1*(G0 + P0*C0) = G1 + P1*G0 + P1*P0*C0

C3 = G2 + P2*C2 = G2 + P2*(G1 + P1*G0 + P1*P0*C0) = G2 + P2*G1 + P2*P1*G0 + P2*P1*P0*C0

C4 = G3 + P3*C3 = G3 + P3*(G2 + P2*G1 + P2*P1*G0 + P2*P1*P0*C0) = G3 + P3*G2 + P3*P2*G1 + P3*P2*P1*G0 + P3* P2*P1*P0*C0

This function grows so large that putting in C31 would most likely take up half the page so I won't show that but it can be seen in the schematic and layout portions. Below we have the truth table of this function form the inputs of the Full Adder, A B C, to the output that clucate the carry G and P which then go and output C_{i+1} which then is used with XOR of A XOR B to give us S sum.

Table 1. Adder Carry Look Ahead Logic Truth Table

A_i	B_i	C_i	S_i	G_i	P_i	C_{i+1}
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	1	0	1	0
0	1	1	0	0	1	1
1	0	0	1	0	1	0
1	0	1	0	0	1	1
1	1	0	0	0	0	1
1	1	1	1	1	0	1

Below is the just Carry Look Ahead logic truth table for a Carry Bit 1.

Table 2. Carry Look Ahead Logic

C_i	G_i	P_i	C_{i+1}
0	0	0	0

0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Before we move on we have to look at the Subtractor part of the machine, to Subtract two number one has to perform 2's complement on one of the inputs then add them. Usually a MUX would be a good solution for this however, since the the two inputs are just the inverses of each other a XOR gate will work just as well. Below we see the XNOR gate which we want to use because it's function is F = ((A' AND B') +(A AND B'))' which is a XOR gate, if we were to implement a XOR gate directly we would get the inverse and would require a inverter which we no longer need.

Table 3. XNOR Truth Table

X	Y	F
0	0	1
0	1	0
1	0	0
1	1	1

Below we can see the set up I've created with the XNOR equation to create a Euler's path.

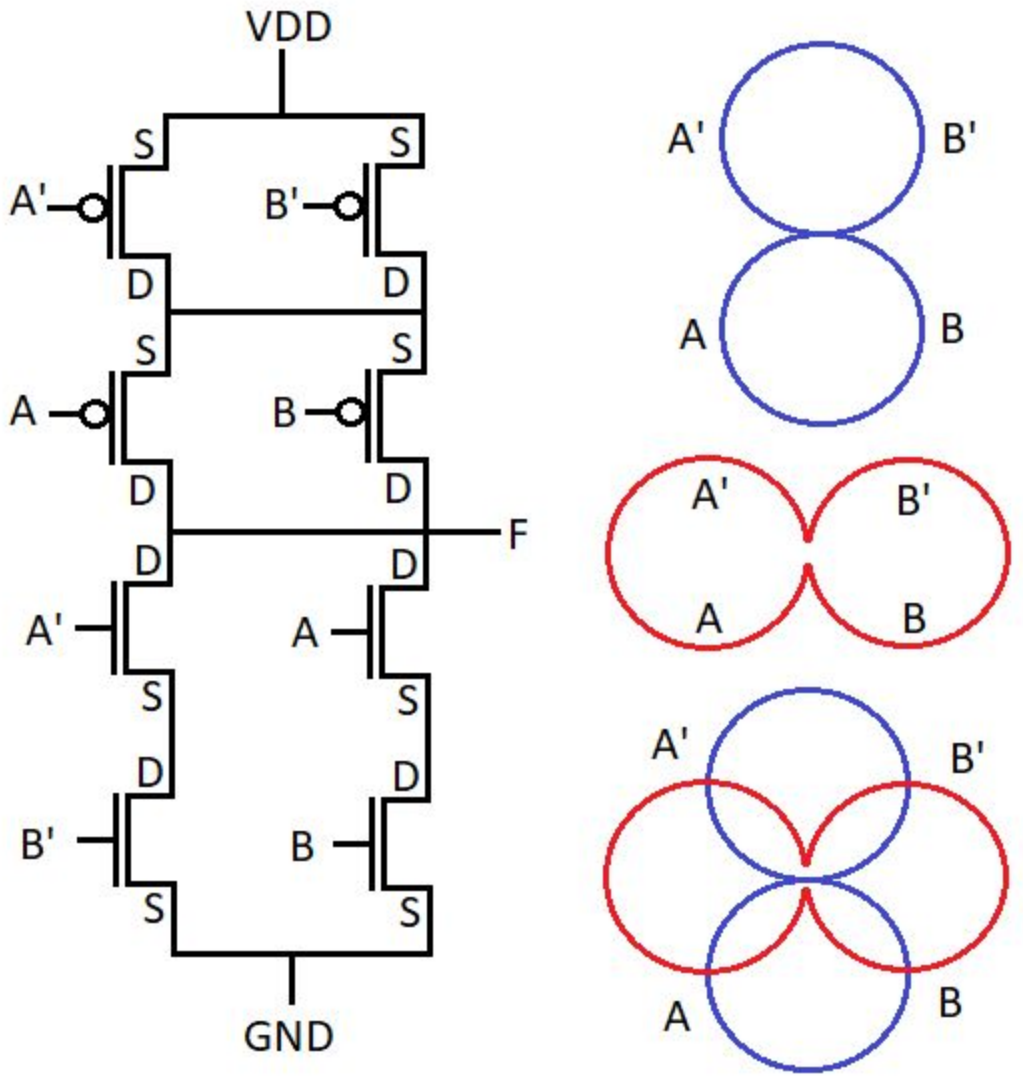


Figure 2. XNOR gate

This setup gives me the following Euler's Paths

AA’B’B, ABB’A’, A’B’BA, A’ABB’, B’BAA’ B’A’AB, BAA’B’ BB’A’A

I chose the B’BAA’ because I planned to set the inverter for B right before the XOR gate on the layout. Figure 8 shows the Stick Diagram for the XOR gate.

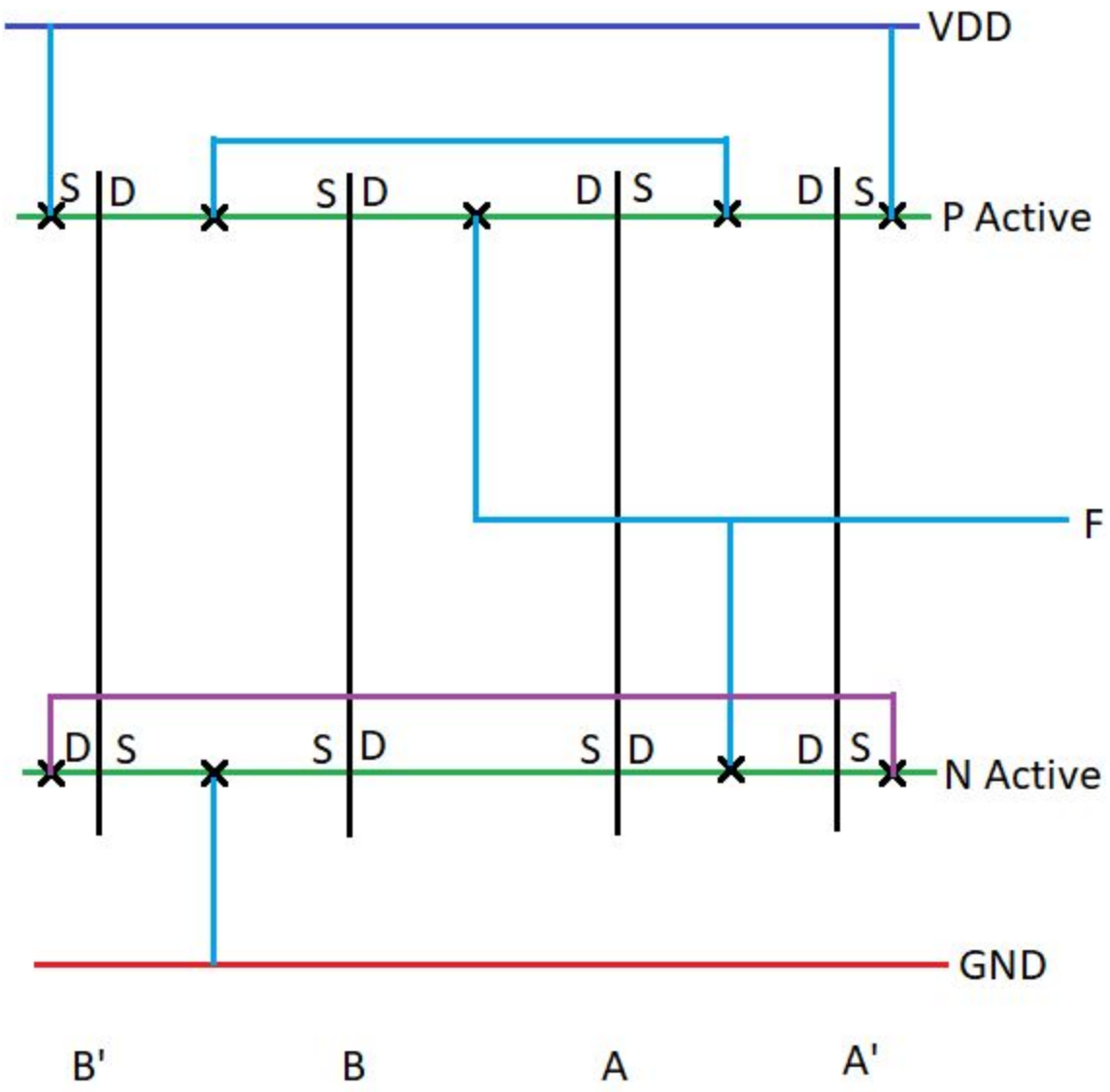


Figure 3. Stick Diagram of XOR gate

All of that together gives us the XOR gate truth table below

Table 4. XOR Truth Table

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0

With that we can create the Truth Table for the Adder/Subtractor with CLA logic, below we have B_new that's the output of the C_0 XOR B_i which allows us to see all the possible G and P that we could obtain.

Table 5. Adder/Subtractor With Carry Look Ahead Logic Truth Table

A_i	B_i	C_0	B_new	C	S	G	P
0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0
0	0	1	1	0	1	0	1
0	0	1	1	1	0	0	1
0	1	0	1	0	1	0	1
0	1	0	1	1	0	0	1
0	1	1	0	0	0	0	0
0	1	1	0	1	1	0	0
1	0	0	0	0	1	0	1
1	0	0	0	1	0	0	1
1	0	1	1	0	0	1	0
1	0	1	1	1	1	1	0
1	1	0	1	0	0	1	0

1	1	0	1	1	1	1	0
1	1	1	0	0	1	0	1
1	1	1	0	1	0	0	1

Section 3: Electric Circuit Schematic

I'm going to go through all of the components I've made in electric, since many of them have 32 bit variations and I don't think it necessary to show every single iteration I will only go through the power of 2, for instance, 2 Input OR gate, 4 Input OR gate, 8 Input OR gate, 16 Input OR gate and 32 Input OR gate.

AND Gates

The CLA requires individual function for each carry to be calculated this means that it requires 32 different gates, in this project I've designed 31 unique gates from 2 AND gate to 32 AND gate. At the start of the project I've worked on creating each AND gate from transistors but as the scale when on I've begun to cascade the existing AND gates to create larger ones something which I didn't do when approaching the layout. In Figures 4 we see 2 Input AND gate, 4 Input AND gate, 8 Input AND gate, 16 Input AND gate and 32 Input AND gate.

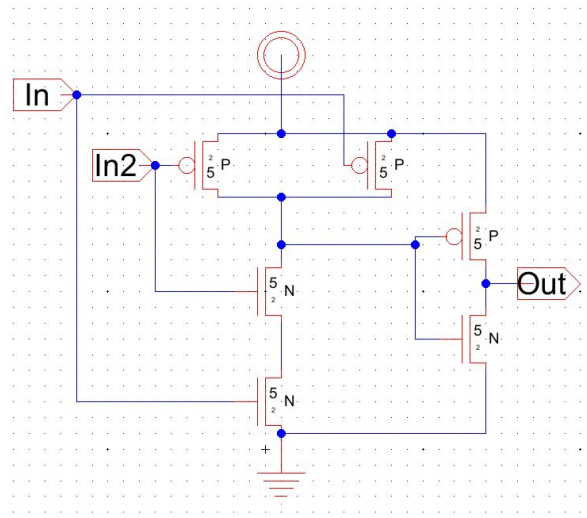


Figure 4.1 2 Input ADD

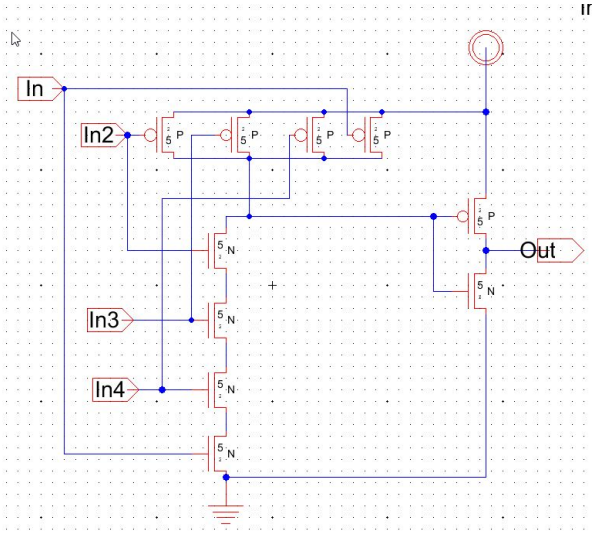


Figure 4.2 4 Input ADD

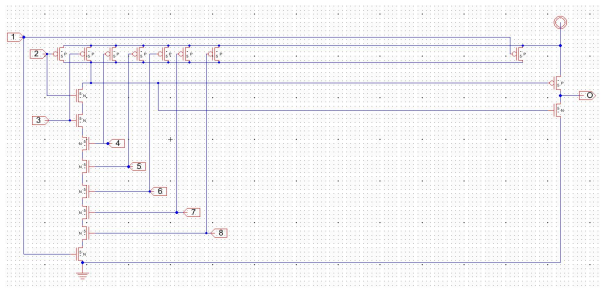


Figure 4.3 8 Input ADD

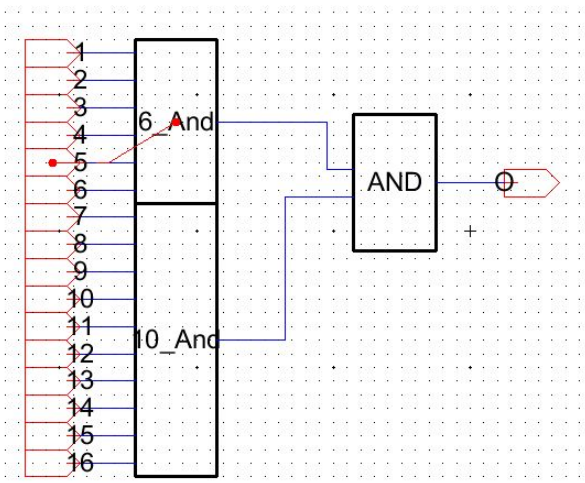


Figure 4.4 16 Input ADD

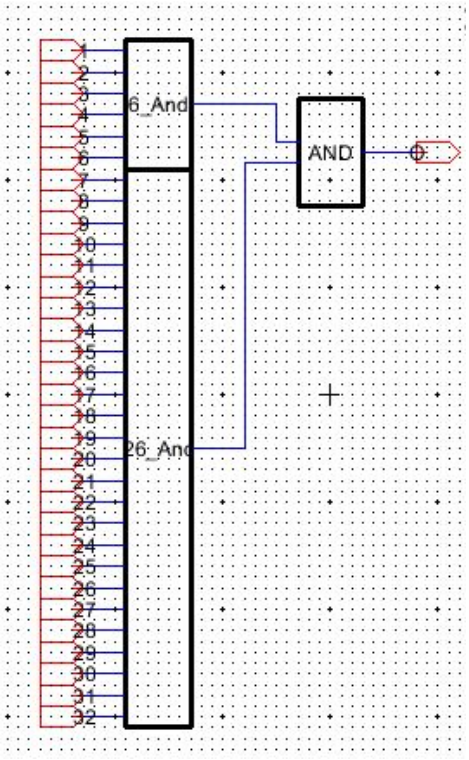


Figure 4.5 32 Input ADD

OR Gates

When the CLA wants to output a signal it looks at the AND gates for all of the different signal variations and sums them up inside an OR gate, this means that I had to create 31 individual OR gates to be able to sum up all the different CLA signals. Similarly to the AND gates I first begun by constructing individual OR gates from transistors but then ended up cascading them to build

larger ones. Below in Figures 5 we can see 2 Input OR gate, 4 Input OR gate, 8 Input OR gate, 16 Input OR gate and 32 Input OR.

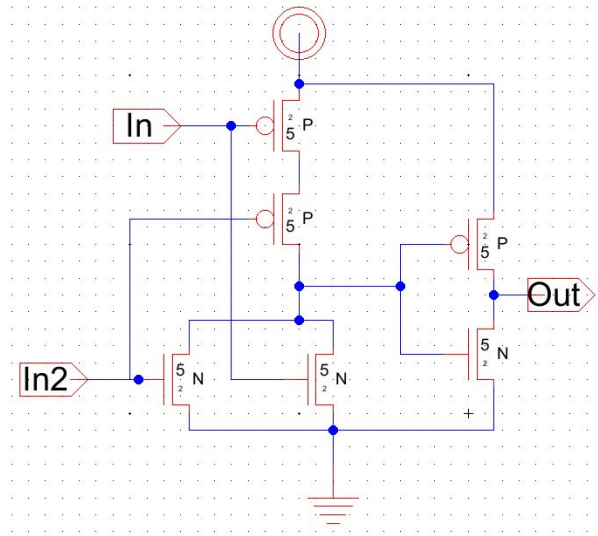


Figure 5.1 2 Input OR

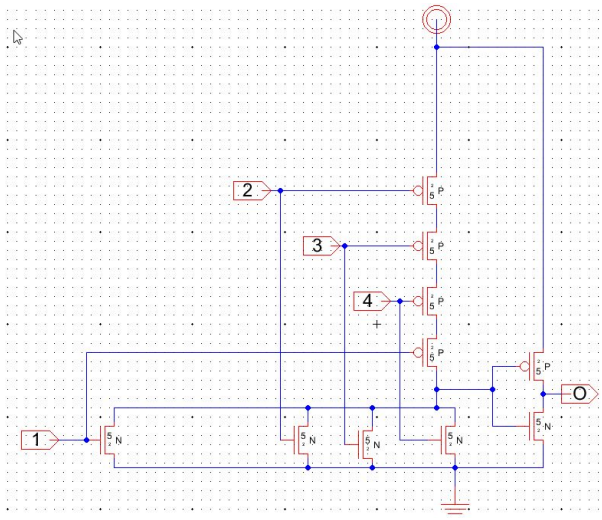


Figure 5.2 4 Input OR

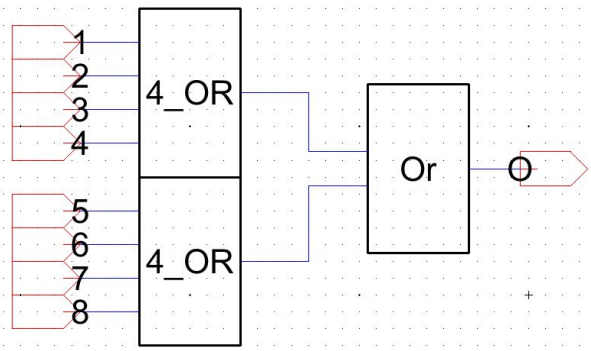


Figure 5.3 8 Input OR

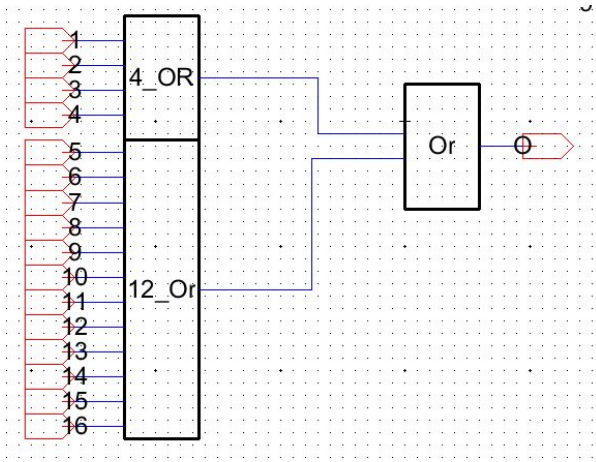


Figure 5.4 16 Input OR

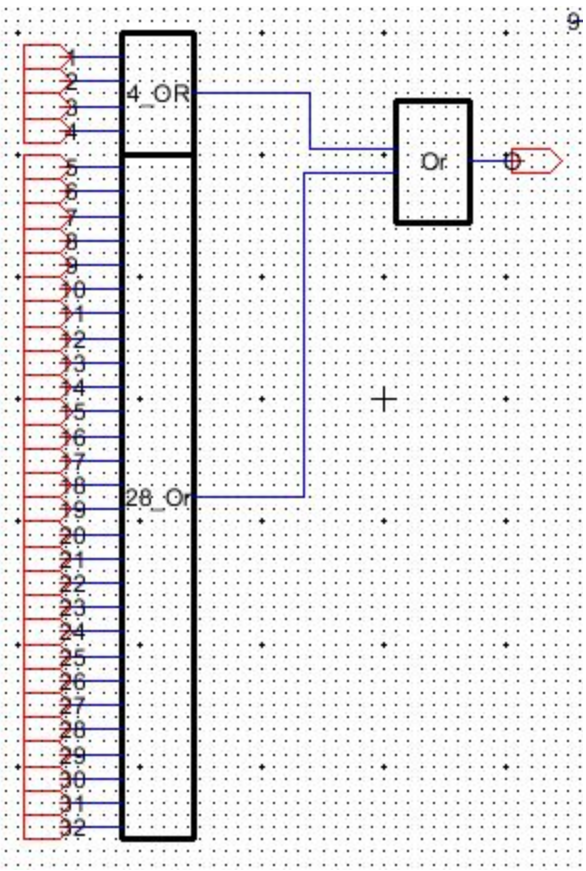


Figure 5.5 32 Input OR

Carry Signals for CLA

For the carry to be correctly it requires the sum of all the Generated Propagation and Propagation, meaning 31 different Carry signals massively all increasing larger in inputs, the way I approached this is by creating the 31 Bit carry signal and then working my way down to Bit 2. As can be seen 31 Bit carry is extremely large on its own and takes in 63 inputs 31 Propagation, 31 Generated Progations and the initial Carry In, every carry signal after that has a unique combination of signals that had to be created, below we can see Carry Bit 2, Carry Bit 4, Carry Bit 8, Carry Bit 16 and Carry Bit 31.

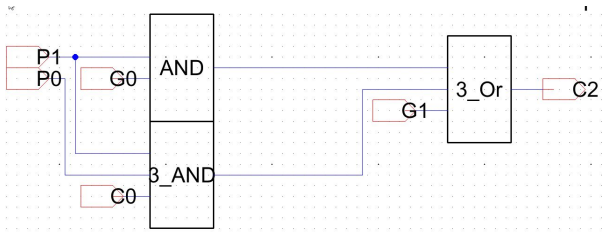


Figure 6.1 Carry Bit 2

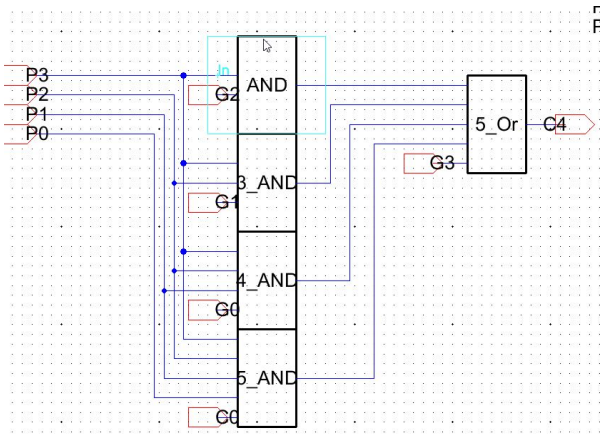


Figure 6.2 Carry Bit 4

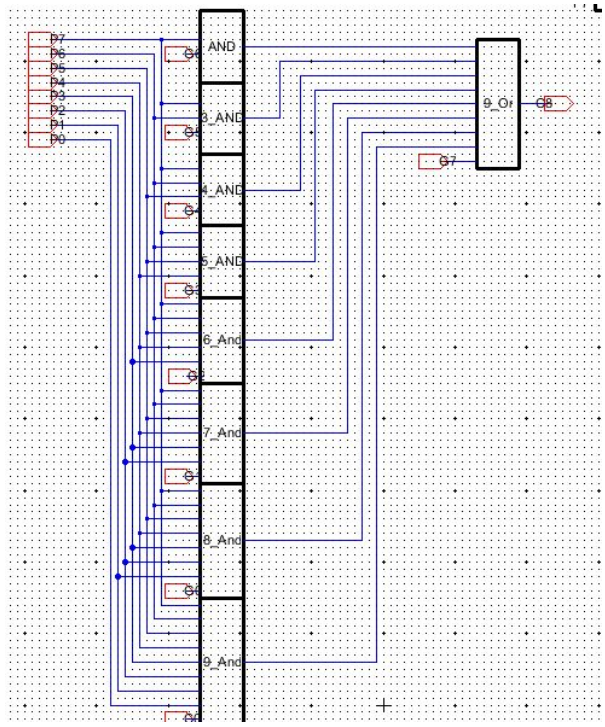


Figure 6.3 Carry Bit 8

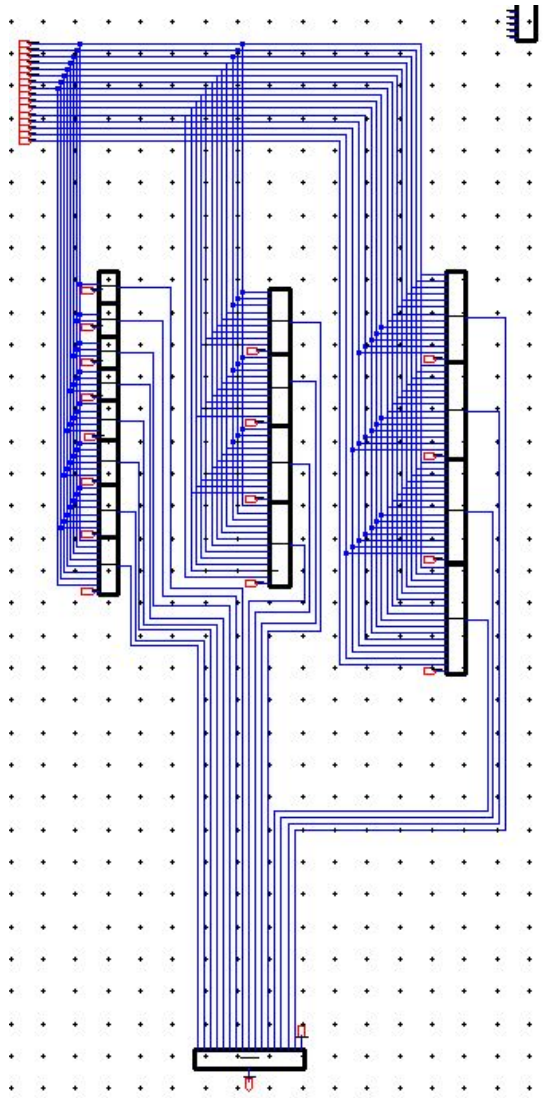


Figure 6.4 Carry Bit 16

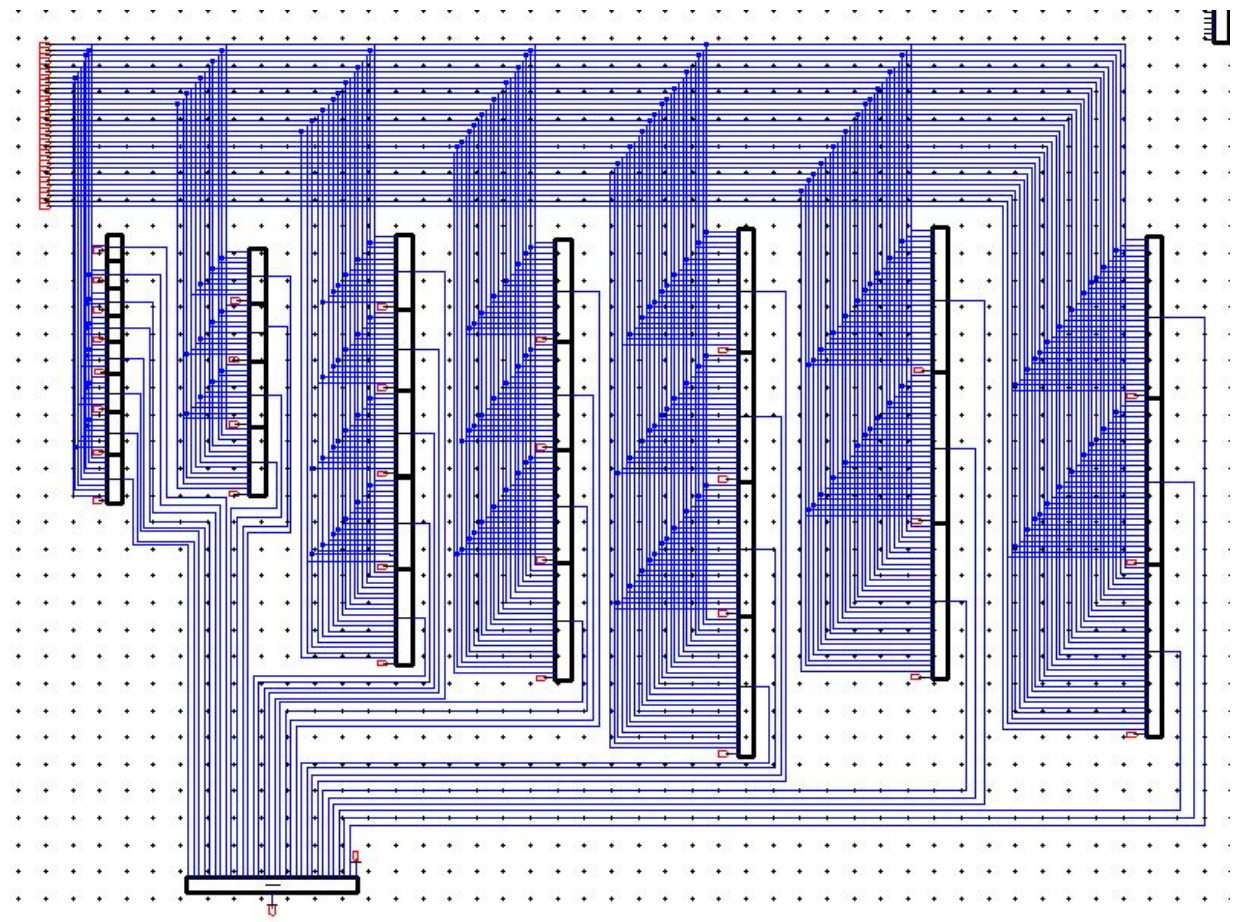


Figure 6.5 Carry Bit 32

CLA

The CLA takes in all 31 of those unique Carry Signals that I've created and contains them inside a box it takes in 63 inputs just like the final Carry Signal and then spreads them out among the 31 components to create a 32 Bit CLA shown below in Figure 7.

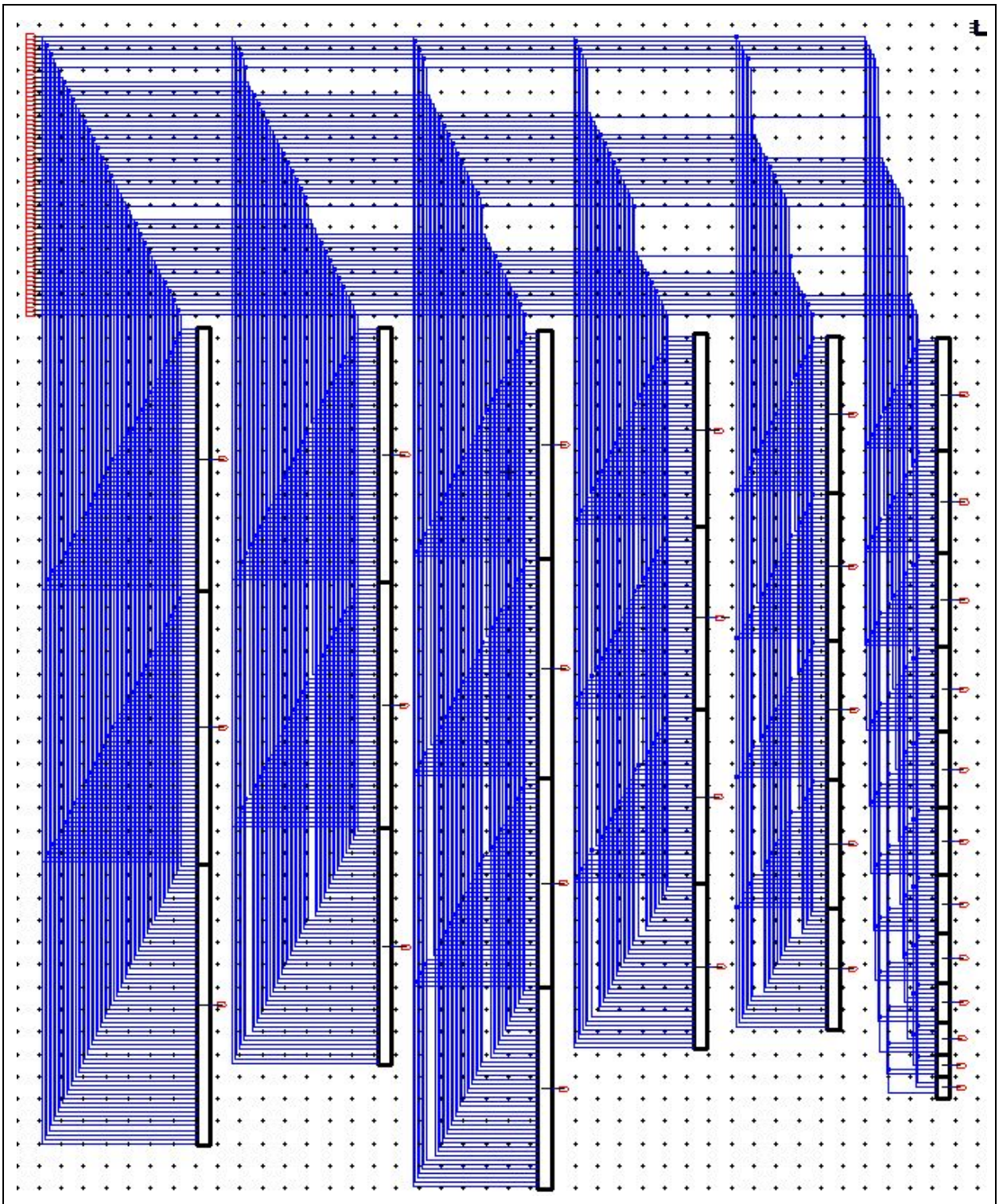


Figure 7 CLA

XOR

The XOR gate is a key component for the Adder Subtractor unit, it allows for the addition of A, B and Carry and allows for the 2's Complement if Carry In is 1. Below is the figure for the XOR gate

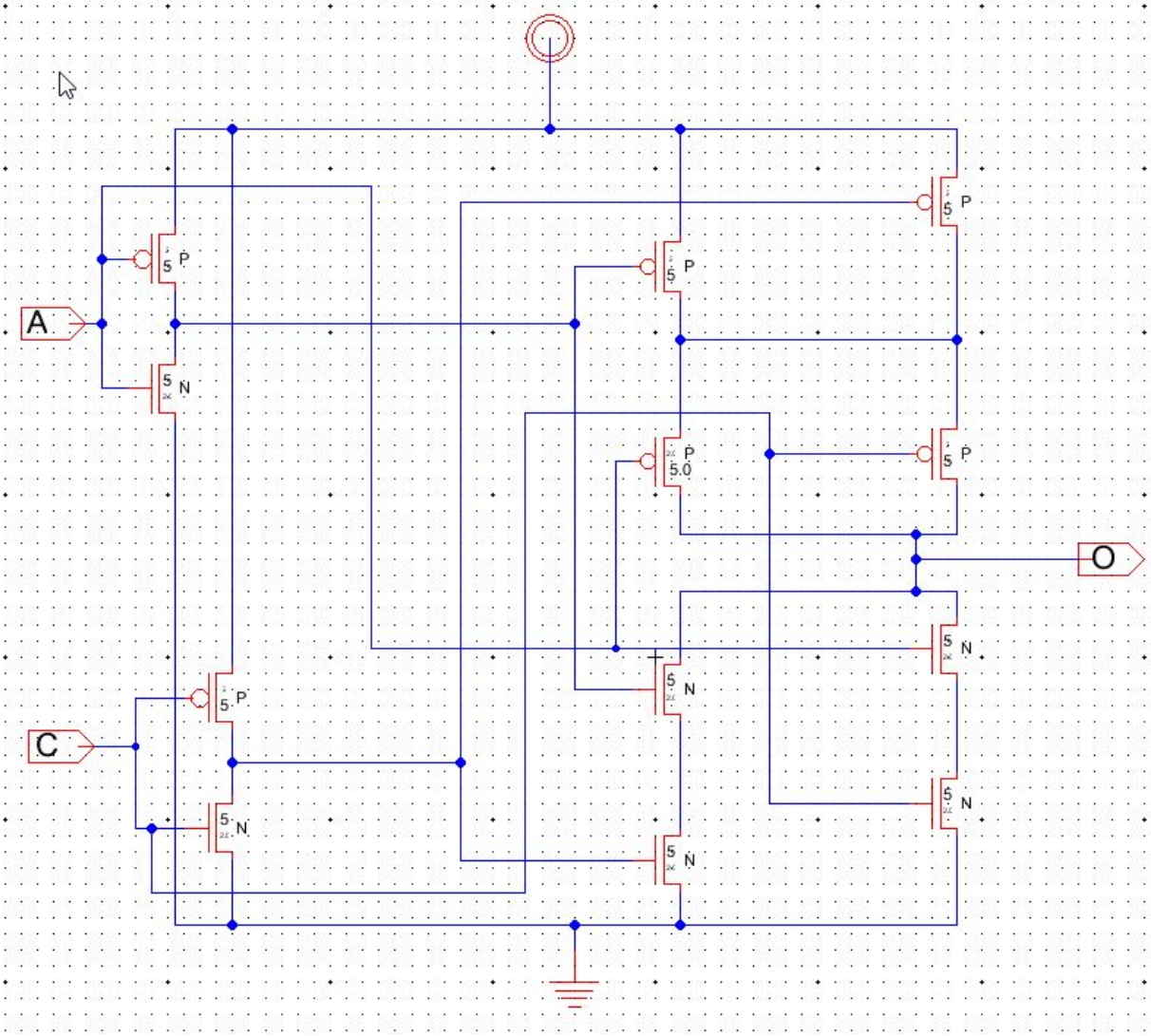


Figure 8 XOR Gate

Full Adder/Subtractor

The Full Adder/Subtractor works based off of 3 XOR gates and 2 Input AND gate, the first XOR gate takes in the inputs of B and C0 which determines if B goes through 2's complement or not. Then the next XOR gate takes in the new B and A to perform addition of the two variables in

that process it creates Propagation (P) output. Meanwhile the same new B and A are ran through a 2 Input AND gate which gives us Generated Propagation (G) output. Lastly the addition of A XOR new B goes is ran through one more XOR gate with the Carry In which results in the Sum (S). This can be seen in the figure below.

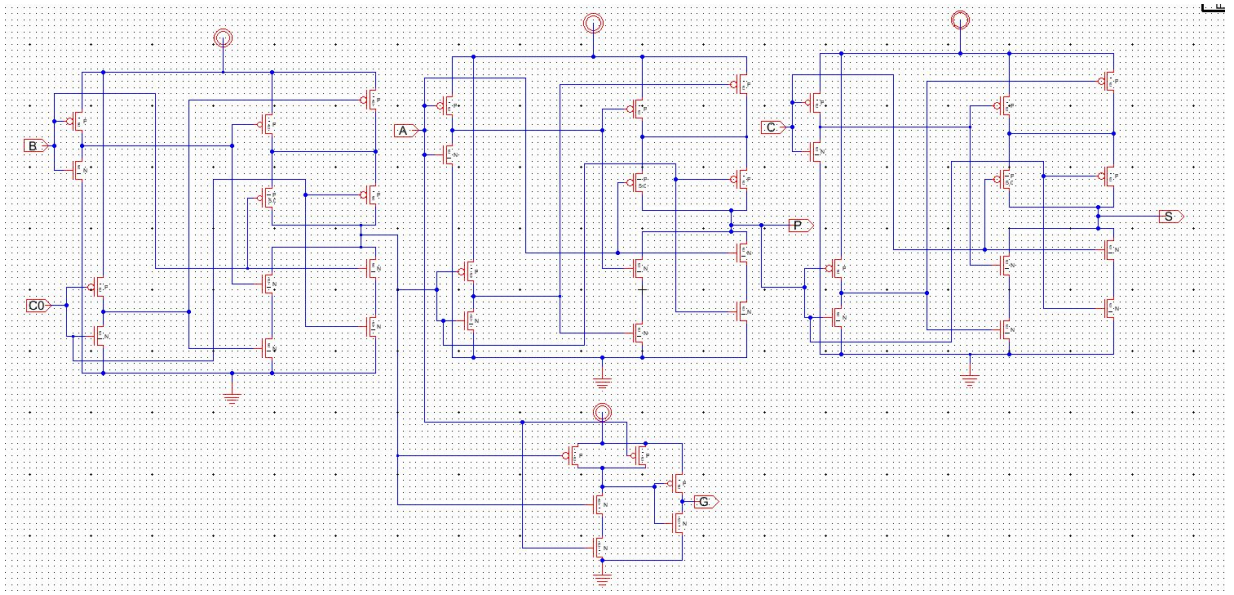


Figure 9 Full Adder Subtractor

32 Bit Adder/Subtractor

Next we take that Full Adder/Subtractor and place it among 31 other adder subtractors, this gives us 96 Inputs, 32 Bits of Number A, 32 Bits of Number B, 31 Carry signals and 1 Carry In that's connected to all of the Full Adder/Subtractors which determines if we are performing Addition or Subtraction and 96 outputs,32 Summations, 32 Generated Propagation and 32 Propagation This can be seen in the Figure below.

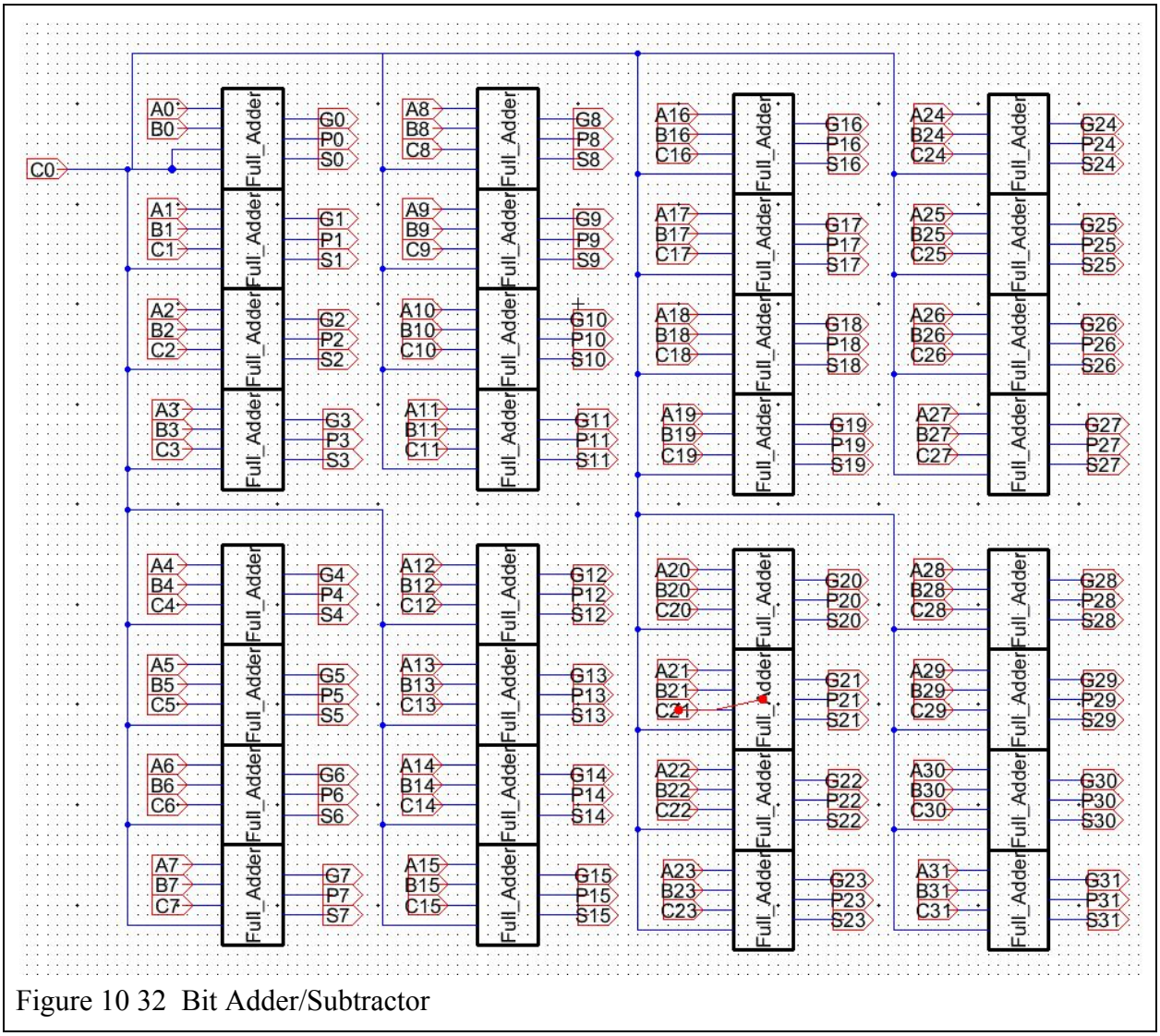


Figure 10 32 Bit Adder/Subtractor

32 Bit Adder/Subtractor Using CLA Logic

Lastly we put together both of the large components of 32 Bit Adder/Subtractor and the 32 Bit CLA to create a fully functioning unit. Below we can see the result which has 96 Inputs same as the ones in used in the 32 Bit Adder/Subtractor and, it has 32 Output which are just the summation and the Generated Propagation and Propagation are plugged into the CLA to calculate the Carry Signals.

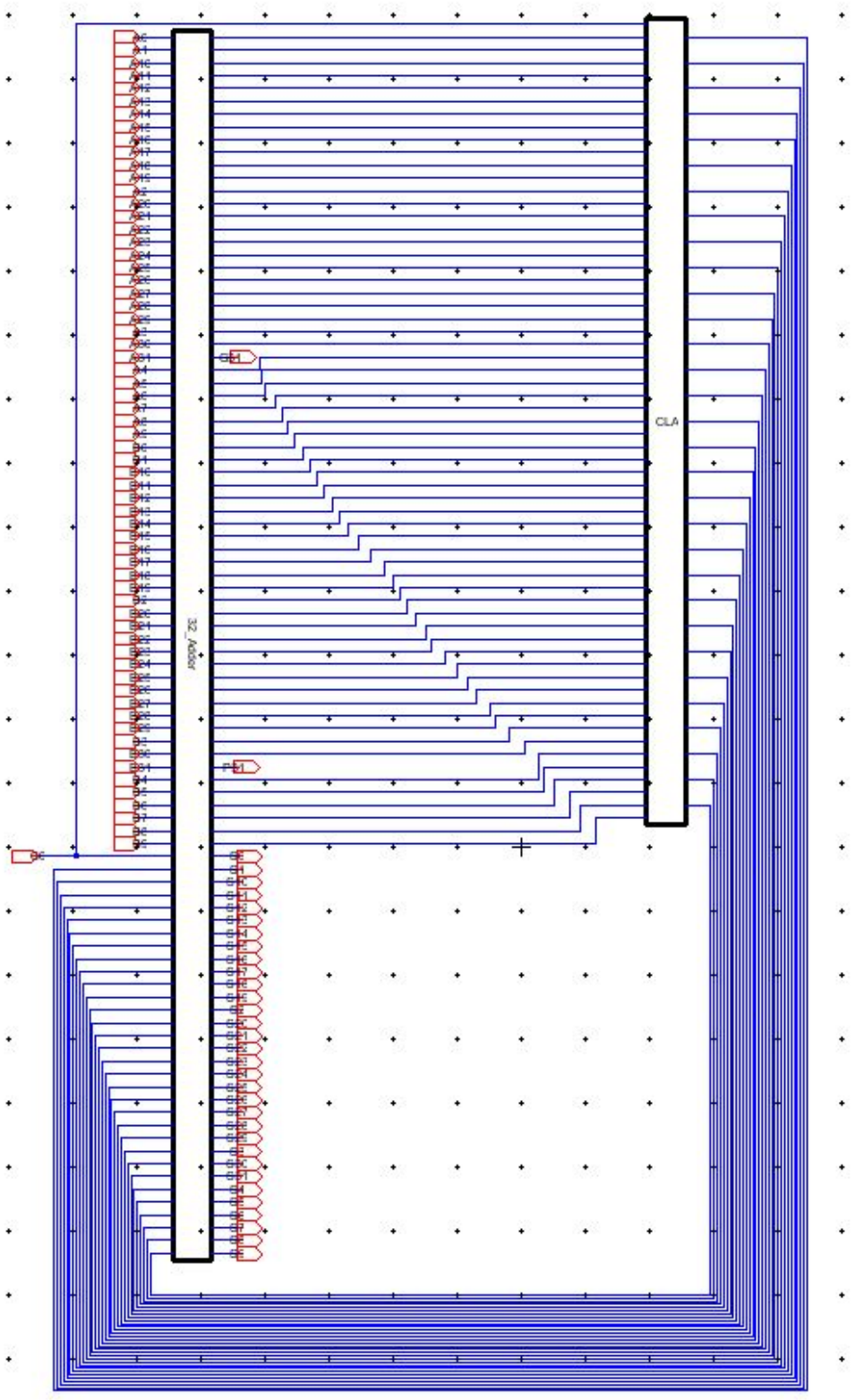


Figure 11 32 Bit Adder/Subtractor Using CLA Logic

DRC

Below the figures show the DRC check which goes through all of the components as we can see there are no issues that Electric can find in any of the files.

```
Checking schematic cell 'Full_Adder{sch}'
   No errors found
Checking schematic cell '32_Adder{sch}'
   No errors found
Checking schematic cell 'C1{sch}'
   No errors found
Checking schematic cell '2_AND{sch}'
   No errors found
Checking schematic cell '3_AND{sch}'
   No errors found
Checking schematic cell '3_Or{sch}'
   No errors found
Checking schematic cell 'C2{sch}'
   No errors found
Checking schematic cell '4_AND{sch}'
   No errors found
Checking schematic cell '4_OR{sch}'
   No errors found
Checking schematic cell 'C3{sch}'
   No errors found
Checking schematic cell '5_AND{sch}'
   No errors found
Checking schematic cell '5_Or{sch}'
   No errors found
Checking schematic cell 'C4{sch}'
   No errors found
Checking schematic cell '6_And{sch}'
   No errors found
Checking schematic cell '6_Or{sch}'
   No errors found
Checking schematic cell 'C5{sch}'
   No errors found
Checking schematic cell '7_And{sch}'
   No errors found
Checking schematic cell '2_Or{sch}'
   No errors found
Checking schematic cell '7_Or{sch}'
   No errors found
Checking schematic cell 'C6{sch}'
   No errors found
Checking schematic cell '8_And{sch}'
   No errors found
Checking schematic cell '8_Or{sch}'
   No errors found
Checking schematic cell 'C7{sch}'
   No errors found
Checking schematic cell '9_And{sch}'
   No errors found
Checking schematic cell '9_Or{sch}'
   No errors found
Checking schematic cell 'C8{sch}'
```

Figure 12.1 Schematic DRC

```
Checking schematic cell 'C7{sch}'
   No errors found
Checking schematic cell '9_And{sch}'
   No errors found
Checking schematic cell '9_Or{sch}'
   No errors found
Checking schematic cell 'C8{sch}'
   No errors found
Checking schematic cell '10_And{sch}'
   No errors found
Checking schematic cell '10_Or{sch}'
   No errors found
Checking schematic cell 'C9{sch}'
   No errors found
Checking schematic cell '11_And{sch}'
   No errors found
Checking schematic cell '11_Or{sch}'
   No errors found
Checking schematic cell 'C10{sch}'
   No errors found
Checking schematic cell '12_And{sch}'
   No errors found
Checking schematic cell '12_Or{sch}'
   No errors found
Checking schematic cell 'C11{sch}'
   No errors found
Checking schematic cell '13_And{sch}'
   No errors found
Checking schematic cell '13_Or{sch}'
   No errors found
Checking schematic cell 'C12{sch}'
   No errors found
Checking schematic cell '14_And{sch}'
   No errors found
Checking schematic cell '14_Or{sch}'
   No errors found
Checking schematic cell 'C13{sch}'
   No errors found
Checking schematic cell '15_And{sch}'
   No errors found
Checking schematic cell '15_OR{sch}'
   No errors found
Checking schematic cell 'C14{sch}'
   No errors found
Checking schematic cell '16_And{sch}'
   No errors found
Checking schematic cell '16_Or{sch}'
   No errors found
Checking schematic cell 'C15{sch}'
   No errors found
Checking schematic cell '17_And{sch}'
```

Figure 12.2 Schematic DRC

```
Checking schematic cell '17_Or{sch}'
   No errors found
Checking schematic cell 'C16{sch}'
   No errors found
Checking schematic cell '18_And{sch}'
   No errors found
Checking schematic cell '18_Or{sch}'
   No errors found
Checking schematic cell 'C17{sch}'
   No errors found
Checking schematic cell '19_And{sch}'
   No errors found
Checking schematic cell '19_Or{sch}'
   No errors found
Checking schematic cell 'C18{sch}'
   No errors found
Checking schematic cell '20_And{sch}'
   No errors found
Checking schematic cell '20_Or{sch}'
   No errors found
Checking schematic cell 'C19{sch}'
   No errors found
Checking schematic cell '21_And{sch}'
   No errors found
Checking schematic cell '21_Or{sch}'
   No errors found
Checking schematic cell 'C20{sch}'
   No errors found
Checking schematic cell '22_And{sch}'
   No errors found
Checking schematic cell '22_Or{sch}'
   No errors found
Checking schematic cell 'C21{sch}'
   No errors found
Checking schematic cell '23_And{sch}'
   No errors found
Checking schematic cell '23_Or{sch}'
   No errors found
Checking schematic cell 'C22{sch}'
   No errors found
Checking schematic cell '24_And{sch}'
   No errors found
Checking schematic cell '24_Or{sch}'
   No errors found
Checking schematic cell 'C23{sch}'
   No errors found
Checking schematic cell '25_And{sch}'
   No errors found
Checking schematic cell '25_Or{sch}'
   No errors found
Checking schematic cell 'C24{sch}'
```

Figure 12.3 Schematic DRC

```
   No errors found
Checking schematic cell 'C24{sch}'
   No errors found
Checking schematic cell '26_And{sch}'
   No errors found
Checking schematic cell '26_Or{sch}'
   No errors found
Checking schematic cell 'C25{sch}'
   No errors found
Checking schematic cell '27_And{sch}'
   No errors found
Checking schematic cell '27_Or{sch}'
   No errors found
Checking schematic cell 'C26{sch}'
   No errors found
Checking schematic cell '28_And{sch}'
   No errors found
Checking schematic cell '28_Or{sch}'
   No errors found
Checking schematic cell 'C27{sch}'
   No errors found
Checking schematic cell '29_And{sch}'
   No errors found
Checking schematic cell '29_Or{sch}'
   No errors found
Checking schematic cell 'C28{sch}'
   No errors found
Checking schematic cell '30_And{sch}'
   No errors found
Checking schematic cell '30_Or{sch}'
   No errors found
Checking schematic cell 'C29{sch}'
   No errors found
Checking schematic cell '31_And{sch}'
   No errors found
Checking schematic cell '31_Or{sch}'
   No errors found
Checking schematic cell 'C30{sch}'
   No errors found
Checking schematic cell '32_And{sch}'
   No errors found
Checking schematic cell '32_Or{sch}'
   No errors found
Checking schematic cell 'C31{sch}'
   No errors found
Checking schematic cell 'CLA{sch}'
   No errors found
Checking schematic cell 'CLA_Full{sch}'
   No errors found
0 errors and 0 warnings found (took 0.184 secs)
```

Figure 12.4 Schematic DRC

Section 4: Detailed Electric Layout

I want to start off with that I know that there are much better ways to design the AND, OR and Carry Signal layouts, using Euler's Path many of these Carry Signal components could have been individual done without the use of AND and OR gates however, due to scale and time that's not possible especially given signals such as Carry Out 31 which take in 65 Inputs and require hundreds of transistors to operate correctly. In addition I know there is a lot of space in the Carry In signal that could have been used more efficiently, that I attribute to the functionality of Electric, it is nigh impossible to move large amount of components and wires without a wire or component going over and creating 100s of errors. Similarly I will not be showing all 32 of each component but rather just five examples scaling by factor of two. Such at we will look at in Figures showing 2 Input OR gate, 4 Input OR gate, 8 Input OR gate, 16 Input OR gate and 32 Input OR.

When it came to the choice of what kind of cmos design I was going to use I chose the basic setup we learned at the beginning of the class, transmission gates would make the creation of OR gate obsolete however that would also mean creating an inverter for every input possible which I believe would outweigh the amount of transistors used to create the OR gates. Secondly there is the choice of Dynamic gates which would in fact cut down the amount of transistors used in half plus two times number of components used, which is still a very significant number and would lead to much longer analysis during IRSIM and LTspice simulations so I've chosen not to use this method either.

AND Gates

Unlike the Schematic all of the AND gates in the layout have been created using only transistors and without cascading smaller gates to create larger ones. All of the PMOS are set up in parallel with every wire going to VDD being a point where two Source points meet and every wire going to output being where Drains meet. On the other hand the NMOS is a series where the very first contact is a Source that connect to ground and the very last contact connects to the output. Lastly there is an inverter attached at the very end to change the NAND gate to an AND gate. In Figures we see 2 AND gate, 4 AND gate, 8 AND gate, 16 AND gate and 32 AND. The reason why they are all set up vertically is so that when I connect them into the Carry signals it's easier to wire them up.

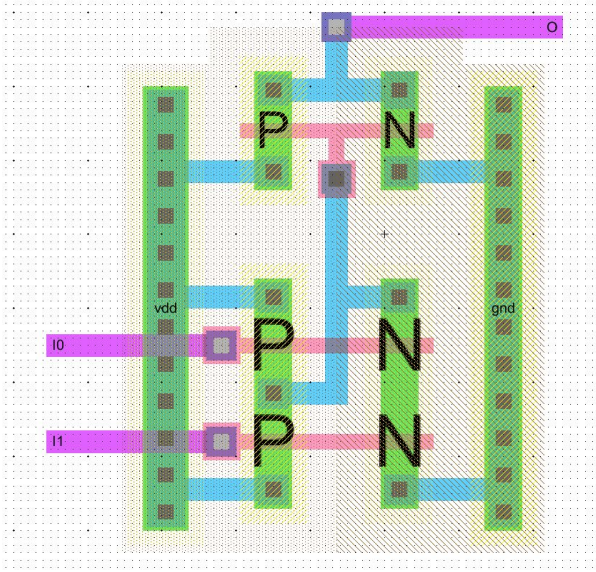


Figure 12.1 2 Input ADD

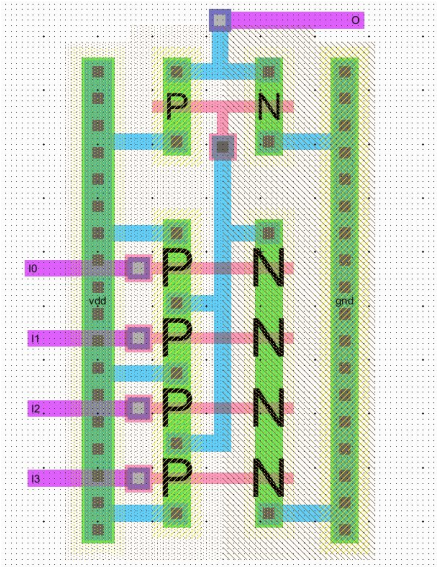


Figure 12.2 4 Input ADD

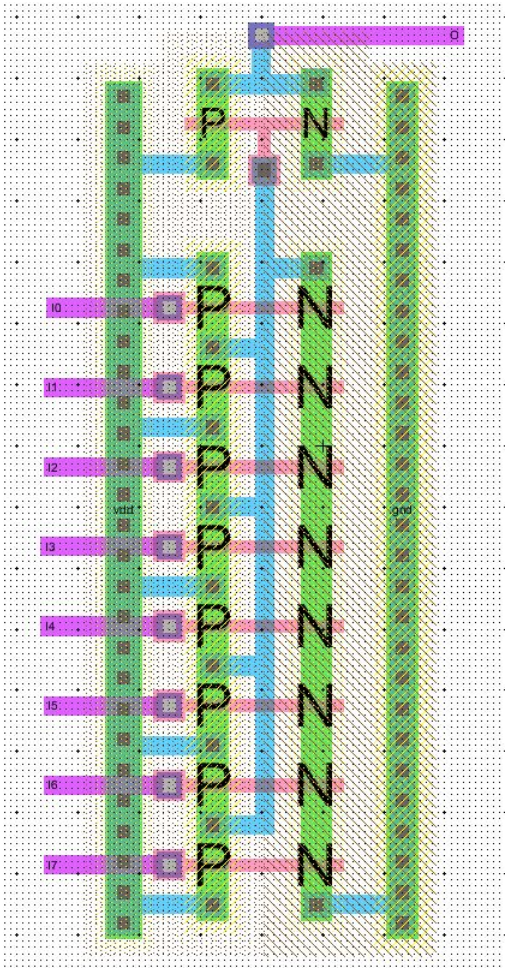


Figure 12.3 8 Input ADD

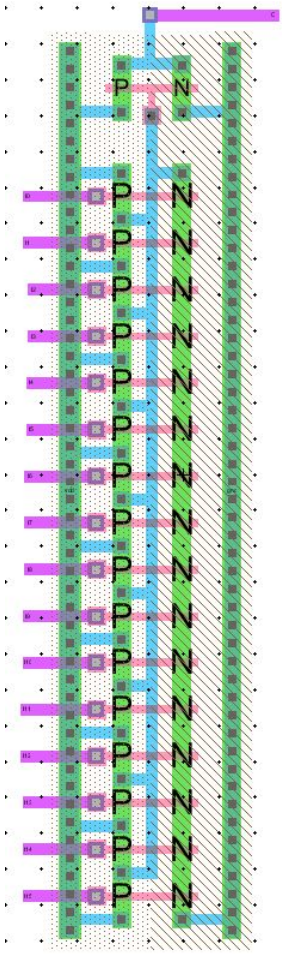


Figure 12.4 16 Input ADD

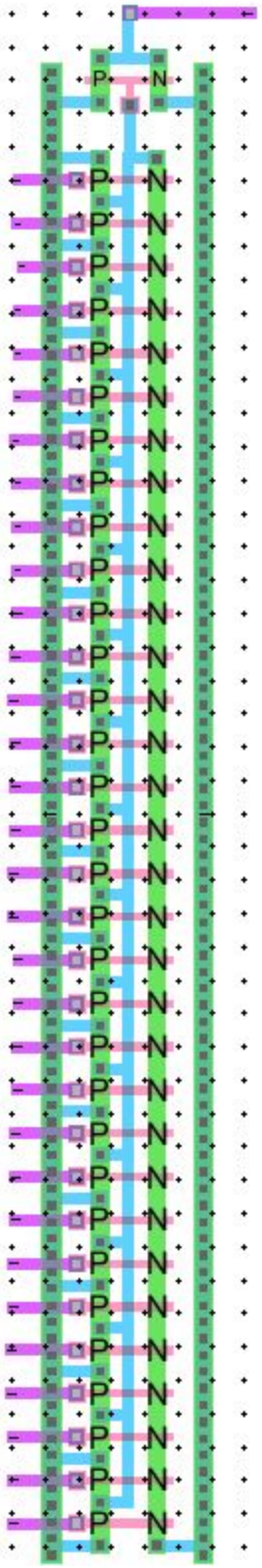


Figure 12.5 32 Input ADD

OR Gates

Similarly here I've created all of the OR gates using transistors going from a 2 Input OR gate all the way to a 32 Input OR Gate. Here we see that the OR gate is just the inverse of the AND gate with the NMOS set up in parallel and the PMOS setup in series. Similarly this one also has a inverter at the very end to change the output from a NOR gate to and OR gate. In Figures we see 2 OR gate, 4 OR gate, 8 OR gate, 16 OR gate and 32 OR . Here I've left the layouts as horizontal because they are better used to connect to the multitudes of the AND gate in a such a fashion as will be seen in the next section.

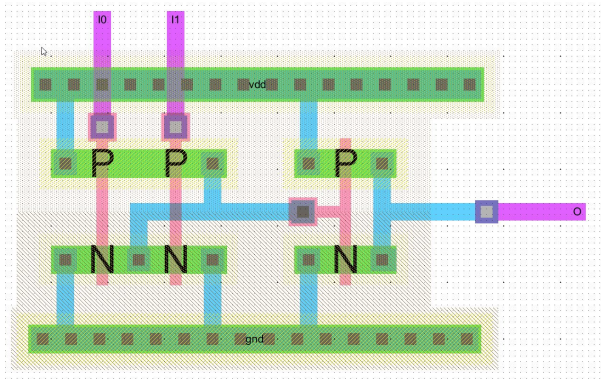


Figure 13.1 2 Input OR

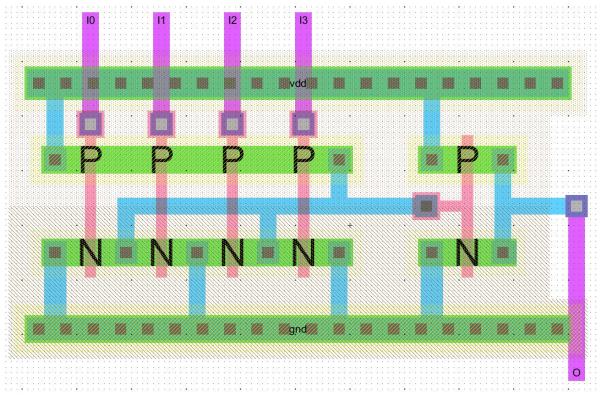


Figure 13.2 4 Input OR

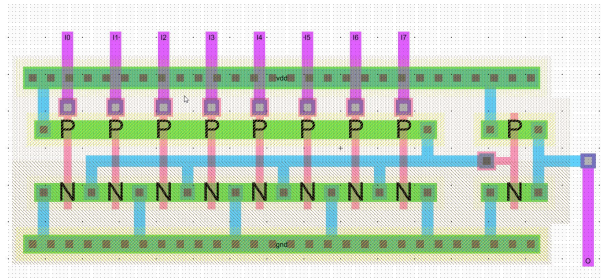


Figure 13.3 8 Input OR

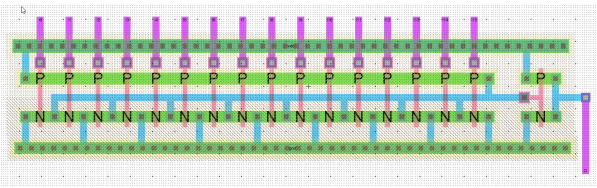


Figure 13.4 16 Input OR

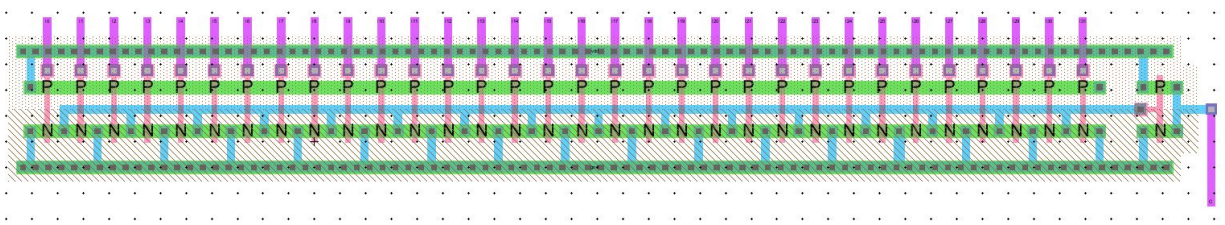


Figure 13.5 32 Input OR

Carry Signals for CLA

Here is where the major work of the project comes in, creating the individual carry signals is a lot of work, similar to the schematic I've started with the largest one the 31 Bit Carry and worked my way down. As mentioned before I know this is now the best set up and the spacing between the components isn't ideal, however Electric cannot handle dealing with so many components and creating a compact layout is extremely tedious. In Figures we see Carry Bit 2, Carry Bit 4, Carry Bit 8, Carry Bit 16 and Carry Bit 31. These 31 components were a lot of work to create and it's very easy to mess one up by the mere fact that there are hundreds of wires running through all of them together, not to mention each wire has to be exported with addition of having to connect all of the VDD and GND source and exporting them as well. To make the job a bit easier I tend to make the third layer of wiring (yellow) the Generated Propagation inputs going to the bottom of the layout and the second layer of wires that (purple) the Propagation.

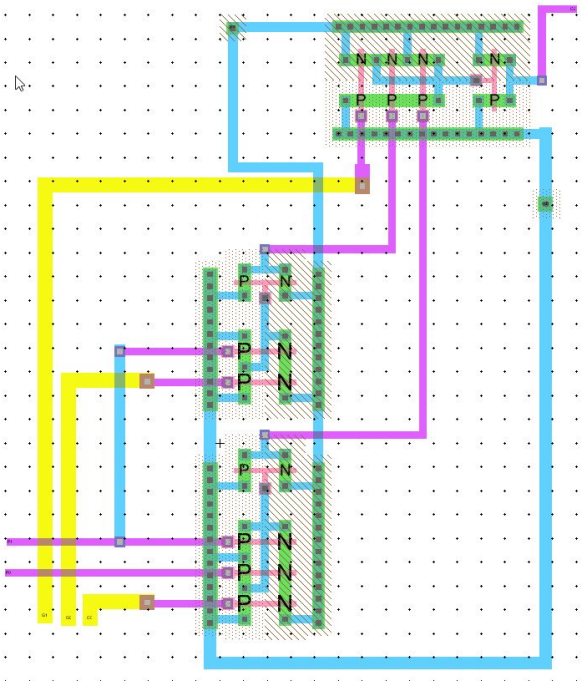


Figure 14.1 Carry Bit 2

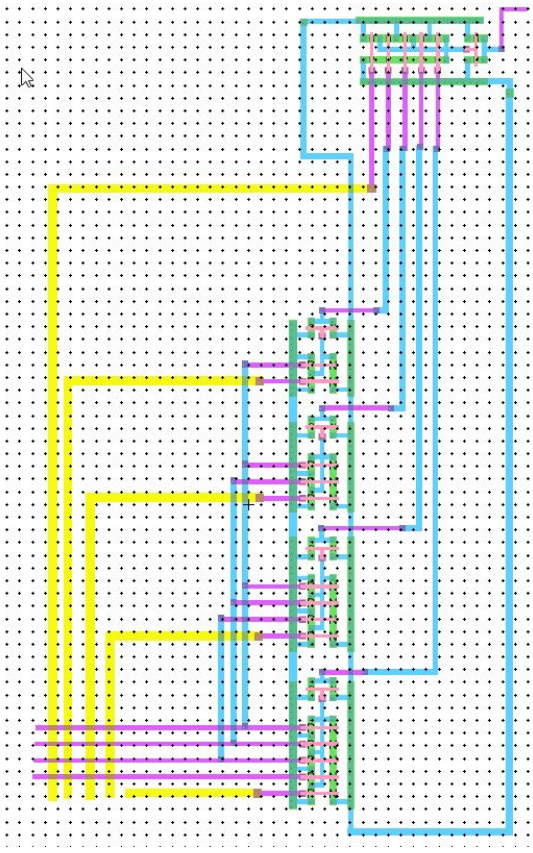


Figure 14.2 Carry Bit 4

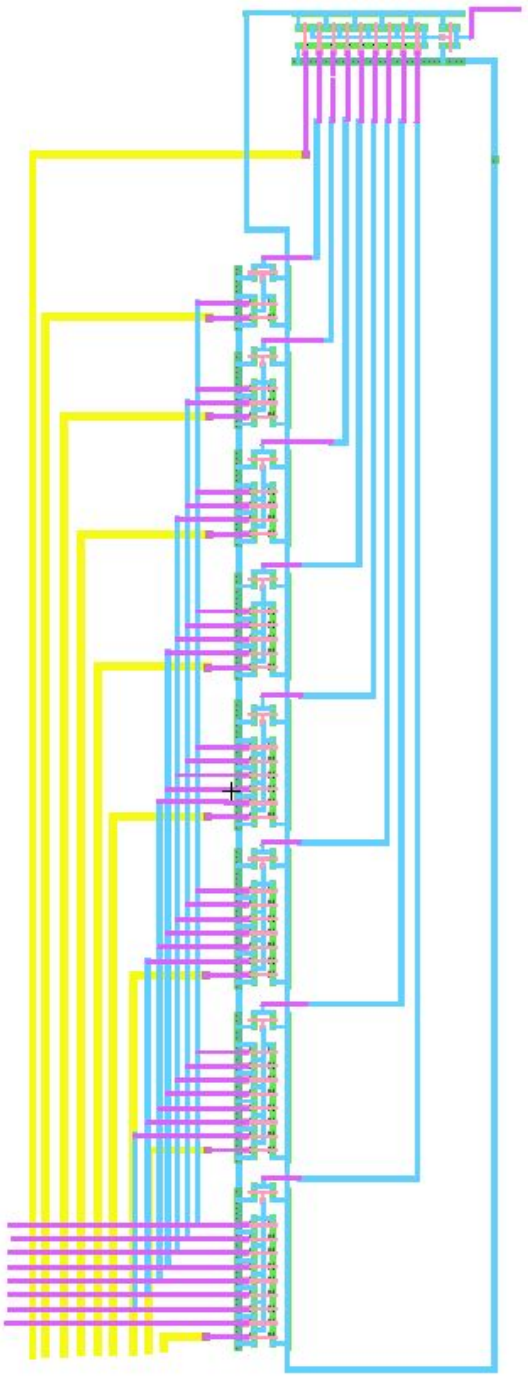


Figure 14.3 Carry Bit 8

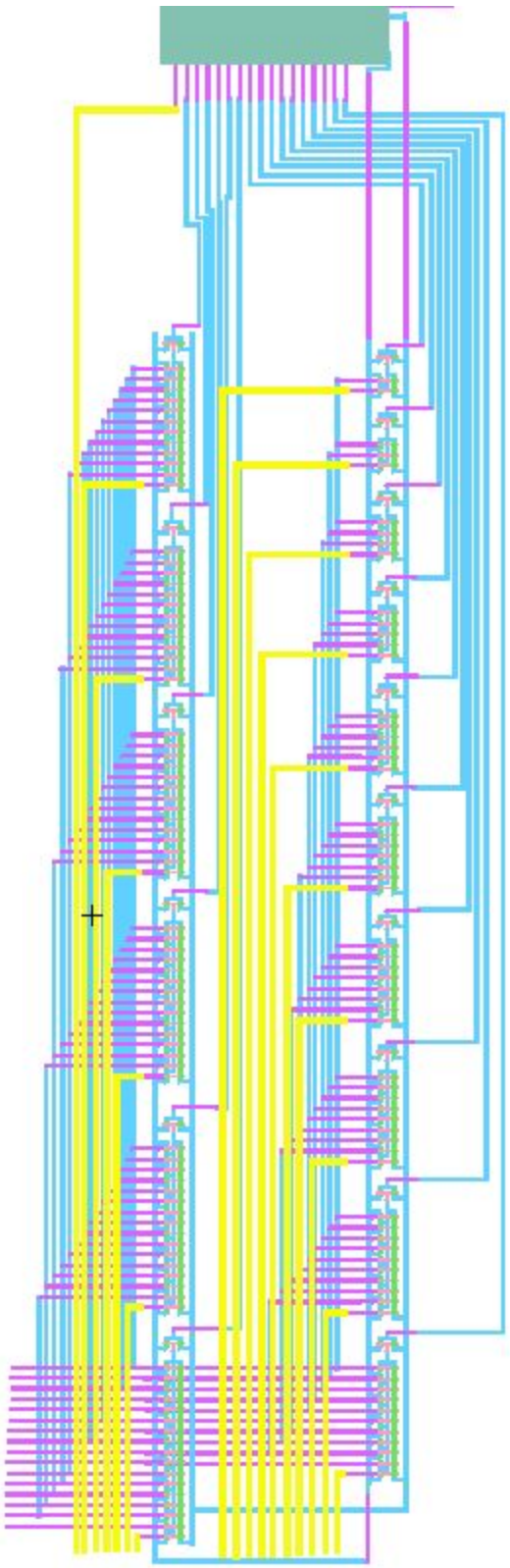


Figure 14.4 Carry Bit 16

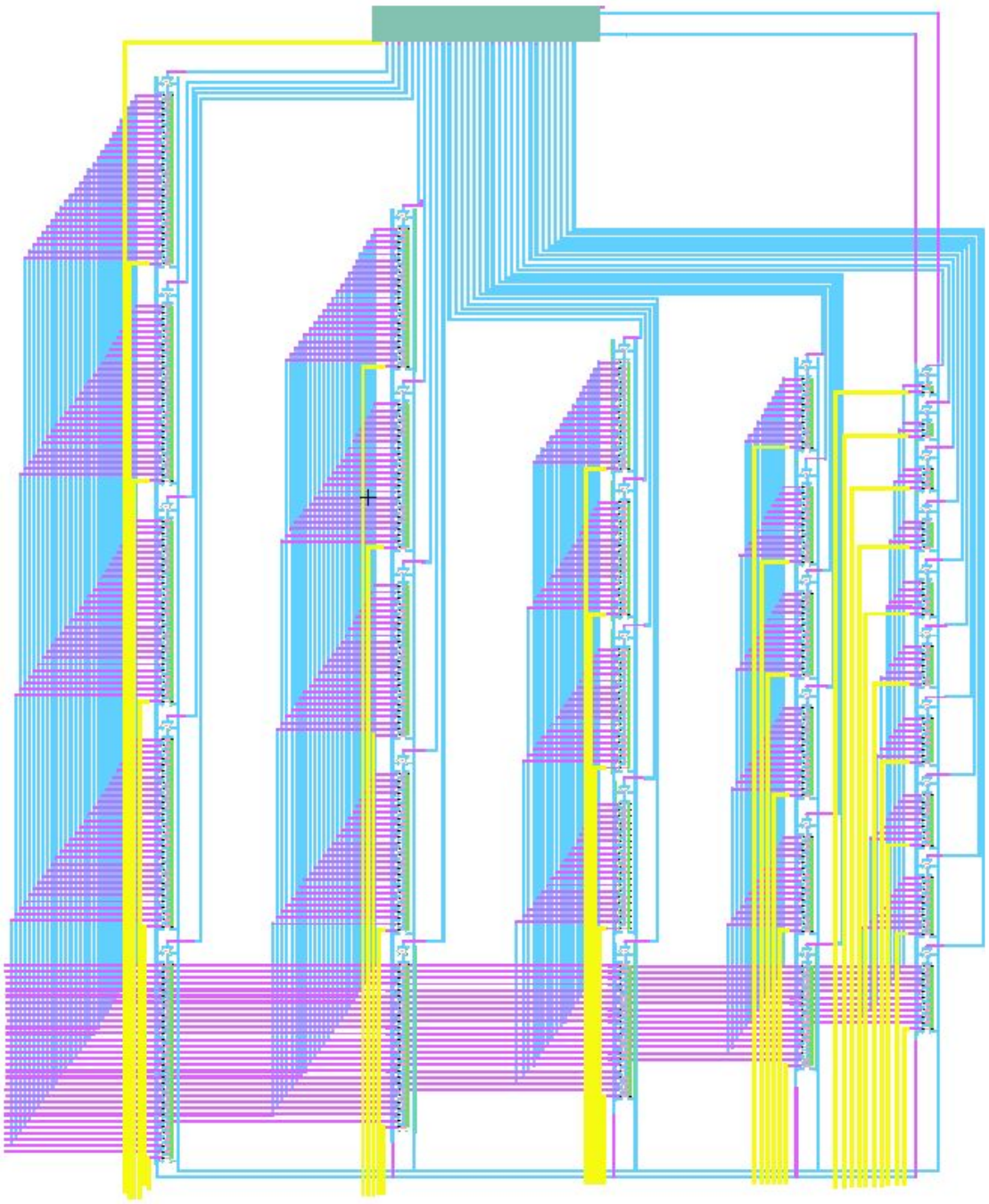


Figure 14.5 Carry Bit 31

CLA

Below is the CLA, I was unable to complete this part of the layout,after testing of many of the components I found that the AND and OR gates I've constructed were not suitable and components such as Carry Bit 31 did done function thus I stopped here.

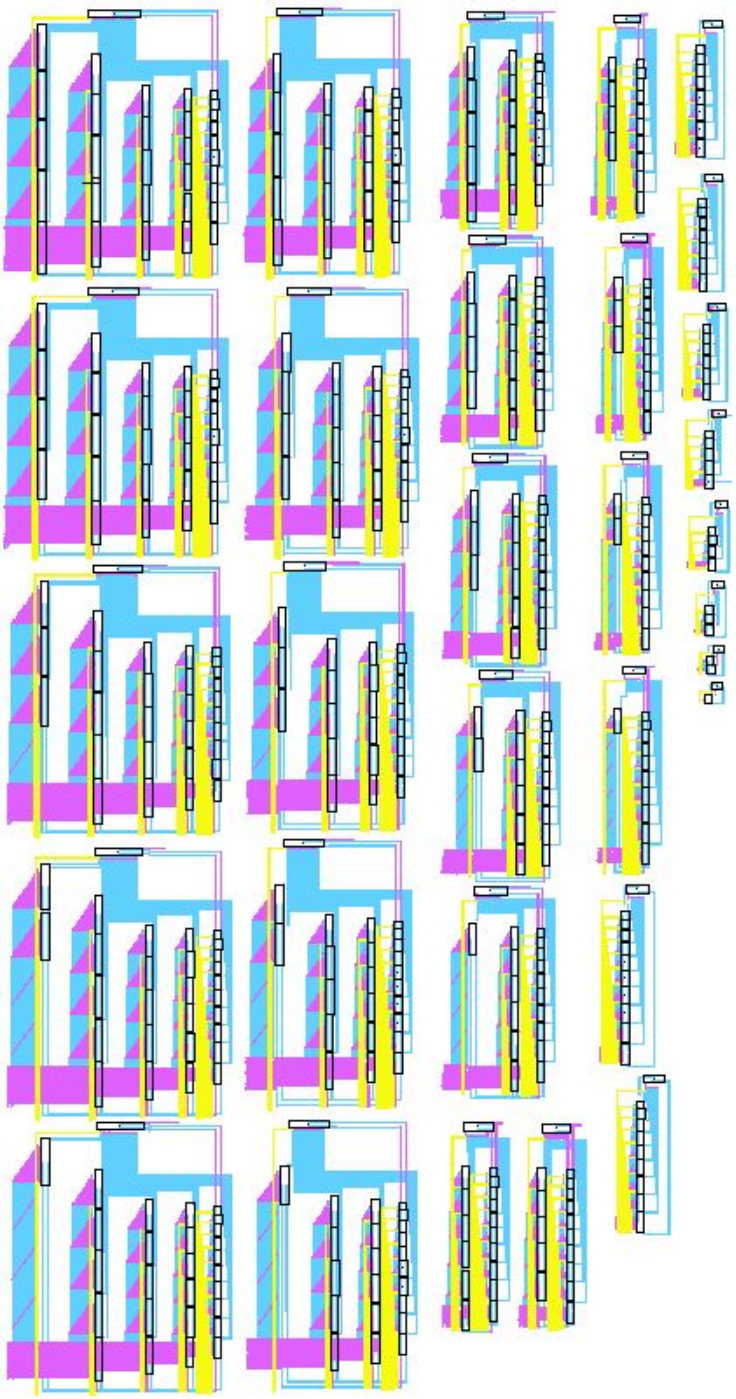


Figure 15 CLA

XOR

The XOR gate as shown below is made up of two inverters and the special logic that was shown in Section 2 Figure . It takes in one bit of input A and B and produces the output of a summation.

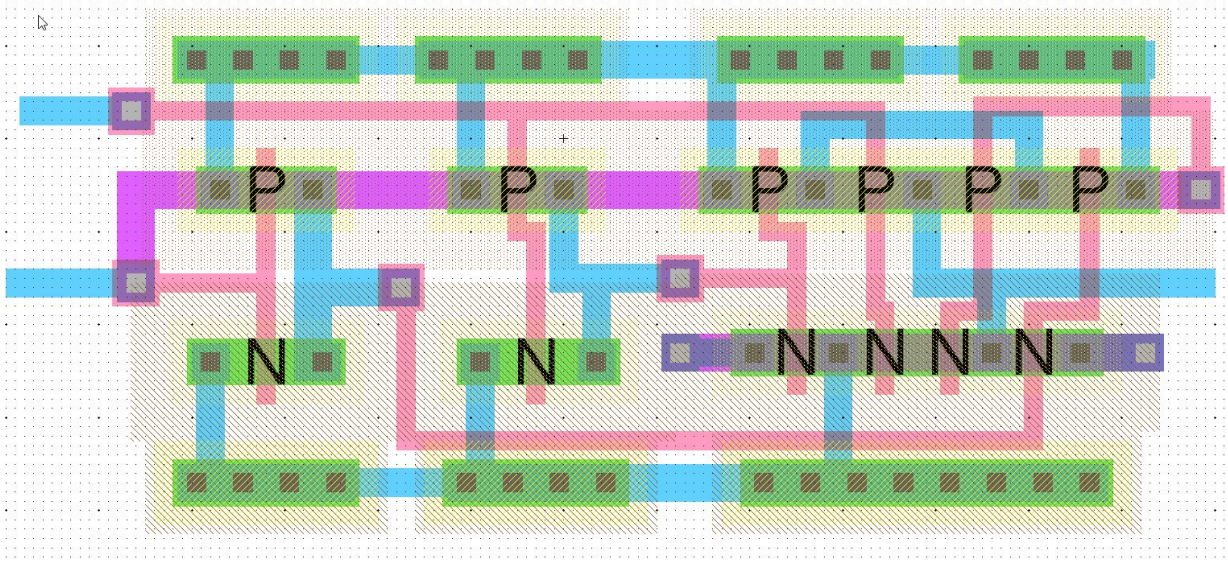


Figure 16 XOR

Full Adder/Subtractor

Below is the Full Adder/Subtractor it works exactly the same as explained in the Schematic section, first we have XOR gate between B and Carry In, then a XOR gate between B' and A this gives out the Propagation signal (P), then finally a XOR gate between the sum of B' XOR A and the carry in which gives out the Sum (S) and the last component being a 2 Input AND gate between B' AND A which gives off Generated Propagation (G). The four inputs on the left are of are A, B, Carry In, Carry, while the outputs on the right are P, S and G.

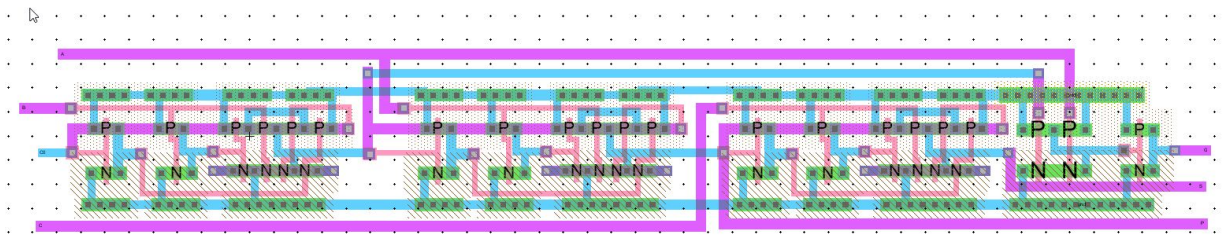


Figure 17 Full Adder/Subtractor

32 Bit Adder/Subtractor

I lined up all of the Full Adder/Subtractor vertically so that they would have easy access to the CLA, as can be seen in the image below.

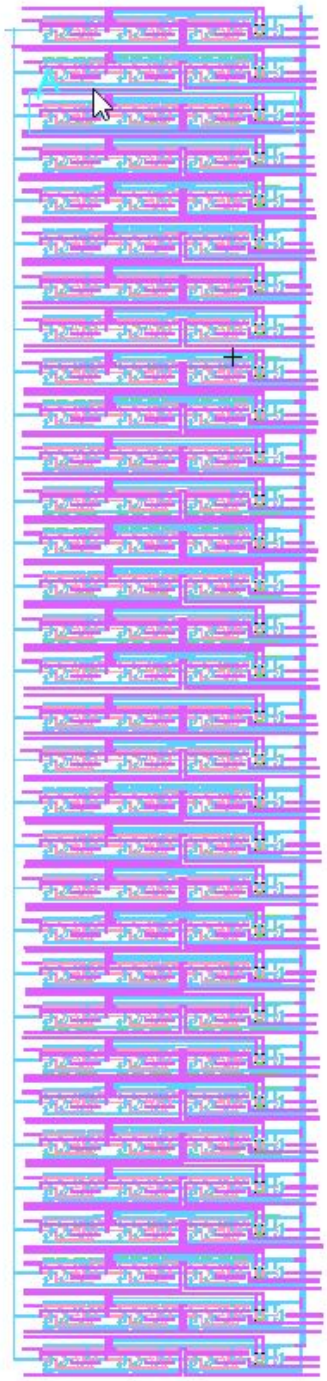


Figure 18 32 Adder/Subtractor

32 Bit Adder/Subtractor Using CLA Logic

Below is the image of what the the 32 Bit Adder/Subtractor next to the CLA, as stated previously its not finished however I've laid them out on the same cell so that I can run a DRC on them.

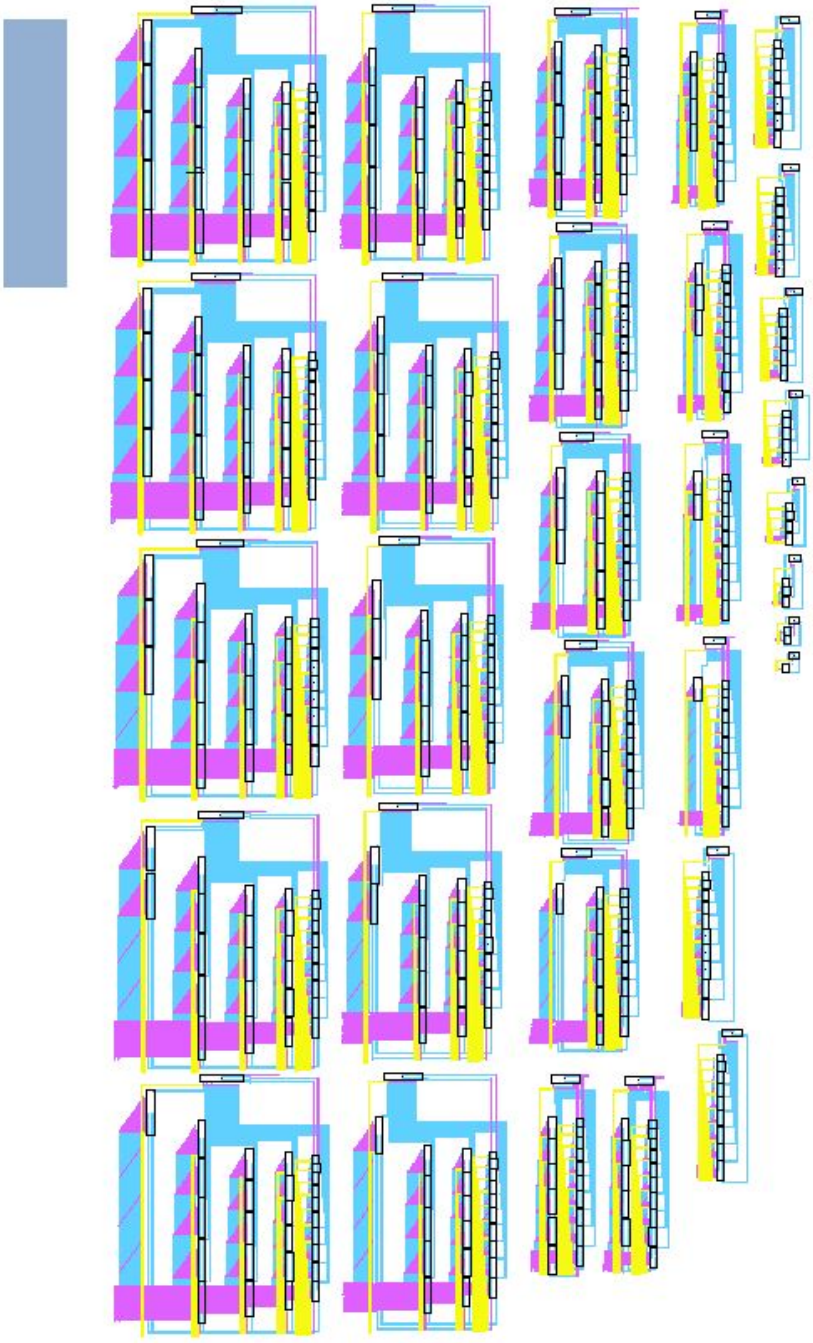


Figure 19 32 Bit Adder/Subtractor Using CLA Logic

DRC

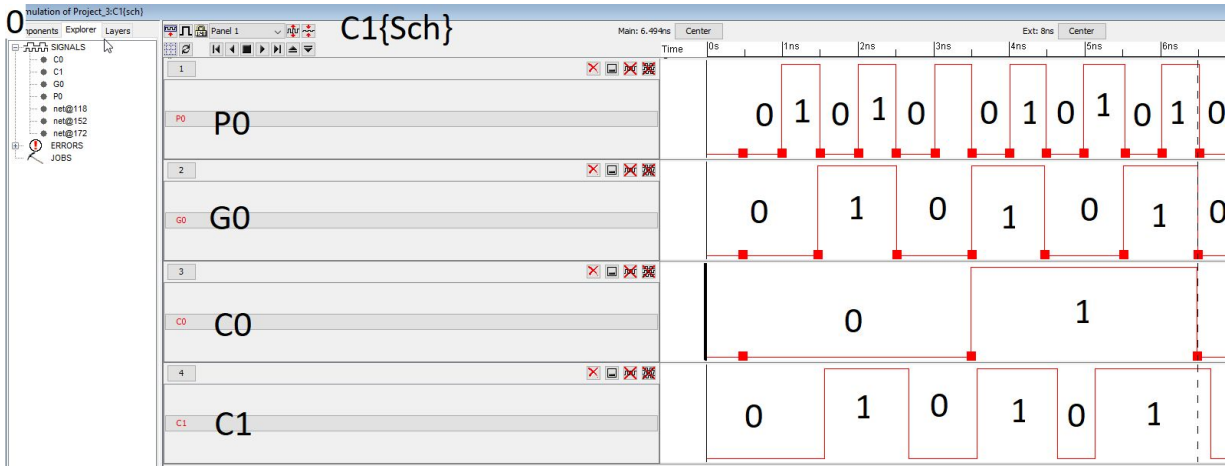
Below is the image of the DRC done on the layout, I’m not sure why it doesn't show me every component like the schematic DRC did but you can see that it took almost 5 minutes for Electric to go through the layout and check every component without giving out any warnings.

```
Running DRC with area bit on, extension bit on, Mosis bit
Checking again hierarchy .... (0.009 secs)
Found 1283 networks
Checking cell 'CLA{lay}'
        No errors/warnings found
0 errors and 0 warnings found (took 4 mins, 38 secs)
Job Design-Rule Check cell 'CLA{lay}' (done) took: 4 mins, 38 secs (started at Sun May 12 13:33:48 EDT 2019, ended at Sun May 12 13:38:27 EDT 2019 )
```

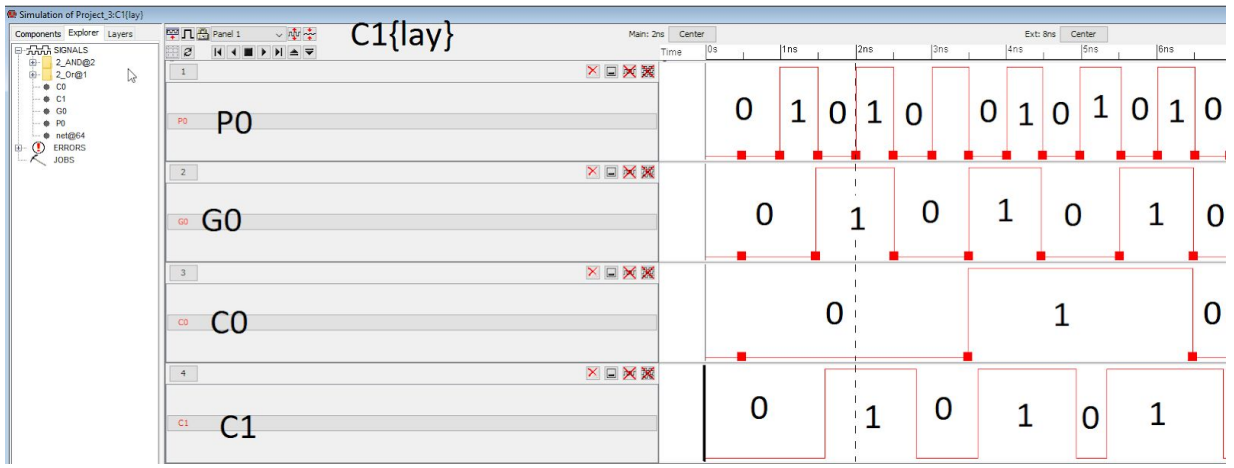
Figure 20 32 Bit Adder/Subtractor Using CLA Logic DRC Layout

Section 5: IRSIM Logic Simulations and Measurements for Layout and Schematic

Since there isn't much I can show when it comes to adding the two 32 Bit numbers I will go through the logic of some of the smaller components and compare them to the tables shown in Section 2. I know there isn't much in this section I ran out of time. The figure below shows the Carry 1 as calculated by P0, G0 and C0 the two sides are almost the same with the layout having a longer delay.



Figue 21. 1 Carry Bit 1 Schematic



Figue 21.2Carry Bit 1 Layout

Section 6: LTSPICE Code and Parasitic Extractions with Analysis and Measurements

This section should be read after Section 7 even though this occurs first in the manual due to the cover sheet requiring it to. Section 7 goes into discovery about the AND and OR gate issues with parasitic components, with this section being a proof for the conclusion that I've came up with in the following section. After compiling Tabel 6 which is in the following section it was clear that the major issue with the system was that the AND and OR gates were performing worst the more inputs they obtained, the OR gates need almost no voltage to turn on and stay of for a considerable amount of time while the AND gates needing a long time to load. Now after performing Parasitic Extraction I can clearly see that the main issue is the way that I positioned the inputs into my AND and OR gates. All of the inputs as shown in Figures shows the area highlighted in black, the major issue with doing such an action resulted in capacitors forming around each output line making the charge and discharge time of each gate much longer than necessary leaving the OR gates in an almost always on state and the AND gates in an almost always off state. As Figure shows the 32 Input OR gate had a total collection of 68.847 fF which way too large than it should be. As for resistors they seem to occur anywhere the poly gets near metal which is almost everywhere as the output runs through almost all of the poly. Overall I could have designed these much better, I didn't think that placing the inputs over the VDD would cause such trouble on large scale.

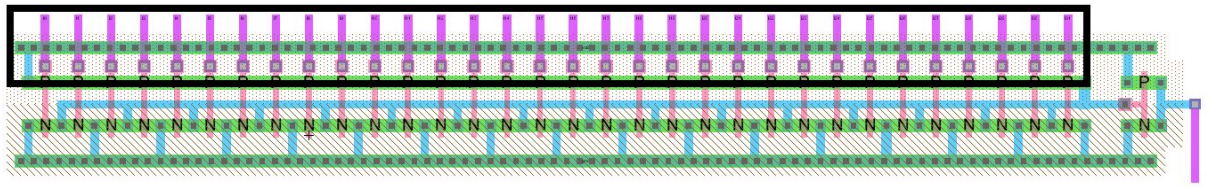


Figure 22. 32 Input OR Gate

```
** Extracted Parasitic Capacitors ***
C0 0 0 7.411fF
C1 net@0#1contact@2_metal-1-polysilicon-1 0 68.847fF
C2 I0 0 1.982fF
C3 I1 0 1.982fF
C4 I2 0 1.982fF
C5 I3 0 1.982fF
C6 I4 0 1.982fF
C7 I5 0 1.982fF
C8 I6 0 1.982fF
C9 I7 0 1.982fF
C10 I8 0 1.982fF
C11 I9 0 1.982fF
C12 I10 0 1.982fF
C13 I11 0 1.982fF
C14 I12 0 1.982fF
C15 I13 0 1.982fF
C16 I14 0 1.982fF
C17 I15 0 1.982fF
C18 I16 0 1.982fF
C19 I17 0 1.982fF
C20 I18 0 1.982fF
C21 I19 0 1.982fF
C22 I20 0 1.982fF
C23 I21 0 1.982fF
C24 I22 0 1.982fF
C25 I23 0 1.982fF
C26 I24 0 1.982fF
C27 I25 0 1.982fF
C28 I26 0 1.982fF
C29 I27 0 1.982fF
C30 I28 0 1.982fF
C31 I29 0 1.982fF
C32 I30 0 1.982fF
C33 I31 0 1.982fF
C34 net@0#2nmos@0_poly-right 0 0.105fF
C35 I0#4nmos@1_poly-left 0 0.157fF
C36 I1#4nmos@2_poly-left 0 0.157fF
C37 I2#4nmos@33_poly-left 0 0.157fF
C38 I3#4nmos@34_poly-left 0 0.157fF
C39 I4#4nmos@35_poly-left 0 0.157fF
C40 I5#4nmos@36_poly-left 0 0.157fF
C41 I6#4nmos@37_poly-left 0 0.157fF
C42 I7#4nmos@38_poly-left 0 0.157fF
C43 I8#4nmos@39_poly-left 0 0.157fF
C44 I9#4nmos@40_poly-left 0 0.157fF
C45 I10#4nmos@41_poly-left 0 0.157fF
C46 I11#4nmos@42_poly-left 0 0.157fF
C47 I12#4nmos@43_poly-left 0 0.157fF
C48 I13#4nmos@44_poly-left 0 0.157fF
C49 I14#4nmos@45_poly-left 0 0.157fF
C50 I15#4nmos@46_poly-left 0 0.157fF
C51 I16#4nmos@47_poly-left 0 0.157fF
C52 I17#4nmos@48_poly-left 0 0.157fF
C53 I18#4nmos@49_poly-left 0 0.157fF
C54 I19#4nmos@50_poly-left 0 0.157fF
C55 I20#4nmos@51_poly-left 0 0.157fF
C56 I21#4nmos@52_poly-left 0 0.157fF
C57 I22#4nmos@53_poly-left 0 0.157fF
C58 I23#4nmos@54_poly-left 0 0.157fF
C59 I24#4nmos@55_poly-left 0 0.157fF
```

Figure 23.1 Capacitor Readings from LTspice

```
C60 I25#4nmos@56_poly-left 0 0.157fF
C61 I26#4nmos@57_poly-left 0 0.157fF
C62 I27#4nmos@58_poly-left 0 0.157fF
C63 I28#4nmos@59_poly-left 0 0.157fF
C64 I29#4nmos@60_poly-left 0 0.157fF
C65 I30#4nmos@61_poly-left 0 0.157fF
C66 I31#4nmos@62_poly-left 0 0.157fF
C67 net@0 0 0.347fF
C68 net@0#3pmos@0_poly-left 0 0.105fF
C69 I0#2pmos@1_poly-left 0 0.255fF
C70 I1#1pmos@3_poly-left 0 0.255fF
C71 I2#2pmos@34_poly-left 0 0.255fF
C72 I3#1pmos@35_poly-left 0 0.255fF
C73 I4#2pmos@36_poly-left 0 0.255fF
C74 I5#1pmos@37_poly-left 0 0.255fF
C75 I6#2pmos@38_poly-left 0 0.255fF
C76 I7#1pmos@39_poly-left 0 0.255fF
C77 I8#2pmos@40_poly-left 0 0.255fF
C78 I9#1pmos@41_poly-left 0 0.255fF
C79 I10#2pmos@42_poly-left 0 0.255fF
C80 I11#1pmos@43_poly-left 0 0.255fF
C81 I12#2pmos@44_poly-left 0 0.255fF
C82 I13#1pmos@45_poly-left 0 0.255fF
C83 I14#2pmos@46_poly-left 0 0.255fF
C84 I15#1pmos@47_poly-left 0 0.255fF
C85 I16#2pmos@48_poly-left 0 0.255fF
C86 I17#1pmos@49_poly-left 0 0.255fF
C87 I18#2pmos@50_poly-left 0 0.255fF
C88 I19#1pmos@51_poly-left 0 0.255fF
C89 I20#2pmos@52_poly-left 0 0.255fF
C90 I21#1pmos@53_poly-left 0 0.255fF
C91 I22#2pmos@54_poly-left 0 0.255fF
C92 I23#1pmos@55_poly-left 0 0.255fF
C93 I24#2pmos@56_poly-left 0 0.255fF
C94 I25#1pmos@57_poly-left 0 0.255fF
C95 I26#2pmos@58_poly-left 0 0.255fF
C96 I27#1pmos@59_poly-left 0 0.255fF
C97 I28#2pmos@60_poly-left 0 0.255fF
C98 I29#1pmos@61_poly-left 0 0.255fF
C99 I30#2pmos@62_poly-left 0 0.255fF
C100 I31#1pmos@63_poly-left 0 0.255fF
```

Figure 23.2 Capacitor Readings from LTspice

```
** Extracted Parasitic Resistors ***
R0 net@0 net@0##0 7.233
C101 net@0##0 0 0.137fF
R1 net@0##0 net@0##1 7.233
C102 net@0##1 0 0.137fF
R2 net@0##1 net@0#1contact@2_metal-1-polysilicon-1 7.233
R3 net@0#2nmos@0_poly-right net@0#2nmos@0_poly-right##0 6.2
C103 net@0#2nmos@0_poly-right##0 0 0.105fF
R4 net@0#2nmos@0_poly-right##0 net@0 6.2
R5 net@0 net@0##0 6.2
C104 net@0##0 0 0.105fF
R6 net@0##0 net@0#3pmos@0_poly-left 6.2
R7 I0#2pmos@1_poly-left I0 7.75
R8 I1#1pmos@3_poly-left I1 7.75
R9 I2#2pmos@34_poly-left I2 7.75
R10 I3#1pmos@35_poly-left I3 7.75
R11 I4#2pmos@36_poly-left I4 7.75
R12 I5#1pmos@37_poly-left I5 7.75
R13 I6#2pmos@38_poly-left I6 7.75
R14 I7#1pmos@39_poly-left I7 7.75
R15 I8#2pmos@40_poly-left I8 7.75
R16 I9#1pmos@41_poly-left I9 7.75
R17 I10#2pmos@42_poly-left I10 7.75
R18 I11#1pmos@43_poly-left I11 7.75
R19 I12#2pmos@44_poly-left I12 7.75
R20 I13#1pmos@45_poly-left I13 7.75
R21 I14#2pmos@46_poly-left I14 7.75
R22 I15#1pmos@47_poly-left I15 7.75
R23 I16#2pmos@48_poly-left I16 7.75
R24 I17#1pmos@49_poly-left I17 7.75
R25 I18#2pmos@50_poly-left I18 7.75
R26 I19#1pmos@51_poly-left I19 7.75
R27 I20#2pmos@52_poly-left I20 7.75
R28 I21#1pmos@53_poly-left I21 7.75
R29 I22#2pmos@54_poly-left I22 7.75
R30 I23#1pmos@55_poly-left I23 7.75
R31 I24#2pmos@56_poly-left I24 7.75
R32 I25#1pmos@57_poly-left I25 7.75
R33 I26#2pmos@58_poly-left I26 7.75
R34 I27#1pmos@59_poly-left I27 7.75
R35 I28#2pmos@60_poly-left I28 7.75
R36 I29#1pmos@61_poly-left I29 7.75
R37 I30#2pmos@62_poly-left I30 7.75
R38 I31#1pmos@63_poly-left I31 7.75
R39 I0#2pmos@1_poly-left I0#2pmos@1_poly-left##0 8.267
C105 I0#2pmos@1_poly-left##0 0 0.157fF
R40 I0#2pmos@1_poly-left##0 I0#2pmos@1_poly-left##1 8.267
C106 I0#2pmos@1_poly-left##1 0 0.157fF
R41 I0#2pmos@1_poly-left##1 I0#4nmos@1_poly-left 8.267
R42 I1#1pmos@3_poly-left I1#1pmos@3_poly-left##0 8.267
C107 I1#1pmos@3_poly-left##0 0 0.157fF
R43 I1#1pmos@3_poly-left##0 I1#1pmos@3_poly-left##1 8.267
C108 I1#1pmos@3_poly-left##1 0 0.157fF
R44 I1#1pmos@3_poly-left##1 I1#4nmos@2_poly-left 8.267
R45 I2#2pmos@34_poly-left I2#2pmos@34_poly-left##0 8.267
C109 I2#2pmos@34_poly-left##0 0 0.157fF
R46 I2#2pmos@34_poly-left##0 I2#2pmos@34_poly-left##1 8.267
C110 I2#2pmos@34_poly-left##1 0 0.157fF
R47 I2#2pmos@34_poly-left##1 I2#4nmos@33_poly-left 8.267
R48 I3#1pmos@35_poly-left I3#1pmos@35_poly-left##0 8.267
C111 I3#1pmos@35_poly-left##0 0 0.157fF
```

Figure 23.3 Resistance Readings from LTspice

Below I included the 32 Input AND gate which is constructed on similar bases as the OR gate and we can see from Figure 25 that it has the same issue that the OR gate does.

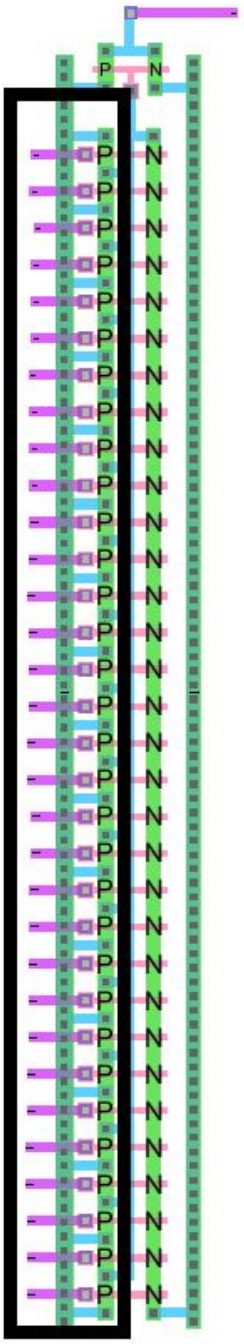


Figure 24. 32 Input AND Gate

```
** Extracted Parasitic Capacitors ***
C0 net@1 0 68.721fF
C1 I31 0 1.982fF
C2 I30 0 2.002fF
C3 I27 0 2.051fF
C4 I28 0 2.051fF
C5 I29 0 2.002fF
C6 I24 0 1.952fF
C7 I25 0 2.002fF
C8 I26 0 2.002fF
C9 I21 0 1.903fF
C10 I22 0 1.952fF
C11 I23 0 1.952fF
C12 I12 0 2.002fF
C13 I13 0 1.952fF
C14 I14 0 1.933fF
C15 I15 0 1.952fF
C16 I16 0 2.002fF
C17 I17 0 2.002fF
C18 I18 0 1.853fF
C19 I19 0 1.853fF
C20 I20 0 1.903fF
C21 I3 0 1.853fF
C22 I4 0 1.853fF
C23 I5 0 1.853fF
C24 I6 0 1.952fF
C25 I7 0 1.903fF
C26 I8 0 1.903fF
C27 I9 0 1.903fF
C28 I10 0 1.903fF
C29 I11 0 1.952fF
C30 I0 0 1.853fF
C31 I1 0 1.903fF
C32 I2 0 1.754fF
C33 0 0 7.472fF
C34 I31#0nmos@0_poly-left 0 0.157fF
C35 I30#4nmos@1_poly-left 0 0.157fF
C36 I29#0nmos@66_poly-left 0 0.157fF
C37 I28#4nmos@67_poly-left 0 0.157fF
C38 I27#0nmos@68_poly-left 0 0.157fF
C39 I26#4nmos@69_poly-left 0 0.157fF
C40 I25#0nmos@70_poly-left 0 0.157fF
C41 I24#4nmos@71_poly-left 0 0.157fF
C42 I23#0nmos@72_poly-left 0 0.157fF
C43 I22#4nmos@73_poly-left 0 0.157fF
C44 I21#0nmos@74_poly-left 0 0.157fF
C45 I20#4nmos@75_poly-left 0 0.157fF
C46 I19#0nmos@76_poly-left 0 0.157fF
C47 I18#4nmos@77_poly-left 0 0.157fF
C48 I17#0nmos@78_poly-left 0 0.157fF
C49 I16#4nmos@79_poly-left 0 0.157fF
C50 I15#0nmos@80_poly-left 0 0.157fF
C51 I14#4nmos@81_poly-left 0 0.157fF
C52 I13#0nmos@82_poly-left 0 0.157fF
C53 I12#4nmos@83_poly-left 0 0.157fF
C54 I11#0nmos@84_poly-left 0 0.157fF
C55 I10#4nmos@85_poly-left 0 0.157fF
C56 I9#0nmos@86_poly-left 0 0.157fF
```

Figure 25 Capacitor Readings from LTspice

Section 7: Measurements in LTSPICE for delays for Layout and Schematic

Here we want to calculate the delay that occurs between the signal input and output, we do that by looking at distance between the two midpoints of the trailing edges of input and output and the two mid points of the leading edges, the sum them and divide them by 2. Since our voltage is at 3.3 V the midpoint we are looking for is at 1.65 V.

We also want to look at the fall and rise time of each of these components so we have to look at when the when the signal is at 10% and 90% for the leading and trailing edges. That being at 2.97 V and 0.33V. I'm going to go through one fully as an example and the rest will be left in Table .

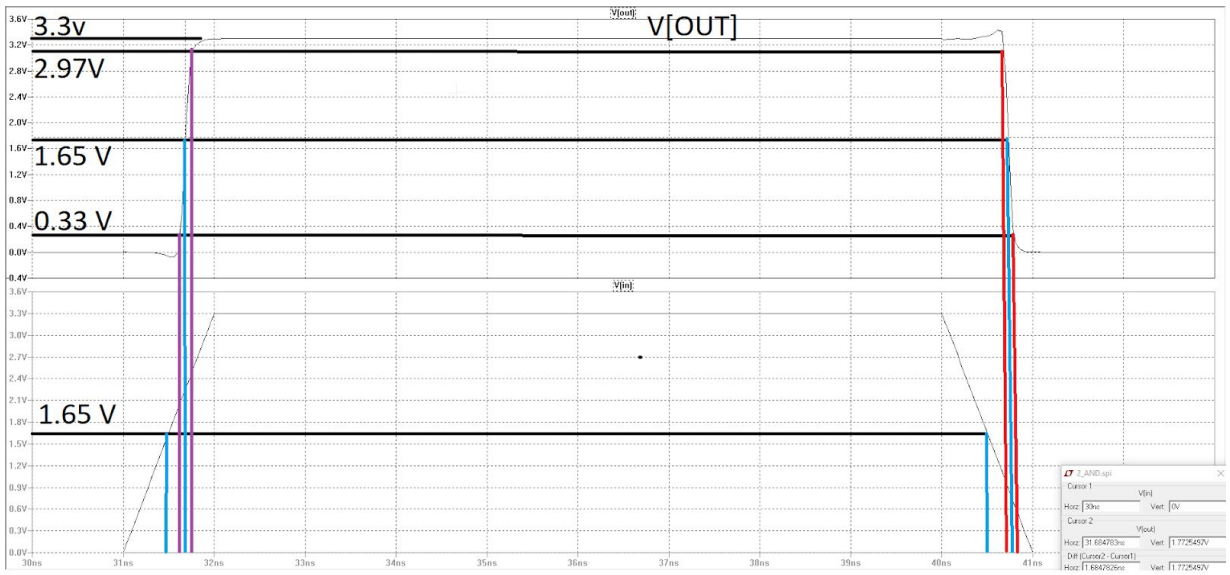


Figure 26 Finding the Delay, Fall Time, Rise Time for 2 Input OR gate

In the image above we see the delays marked by the blue lines, the rising time with the purple lines and the falling time with the red line. Below are the formulas used to calculate the time variables we are looking for.

$$Tr = 31.73ns - 31.625ns = .105ns$$
$$Tf = 40.788587ns - 40.689674ns = 0.098913ns$$
$$tHL = 31.681522ns - 31.511957ns = 0.169565ns$$
$$tLH = 40.73913ns - 40.498913ns = 0.240217ns$$

$$tp = \frac{tLH + tHL}{2} = \frac{0.169565ns + 0.240217ns}{2} = 0.204891ns$$

The way I approached this was for OR, Carry, and Adder I would set all inputs to low and set one input to oscillate. For OR it doesn't matter which input was set to ociliate, however for Carry I made sure to set the current Generated Propagation so that it would go directly to the OR gate for Carry 1 it was G0, and for Carry 31 it was G30 and then I would simply check the output. Similarity for the Adder/Subtractor I set all inputs to low except one input to oscillate, the input I chose was A0 for both Full Adder/Subtractor and 32 Bit Adder/Subtractor and I looked at the output S0 to see the result. Lastly for AND gates I set all signals to high except for one which I made to oscillate.

While going through the entries in the Tabel , I found that the 32 Input AND and 32 Input OR gate layouts had massive disappearances as compared to all the other components they were the most compact units and clearly they had a lot of parasitic parts created when in use, in Figures we can see the large delay for the rising edge for the 32 Input AND Gate and the large delay in the falling on the 32 Input OR gate while the Full Adder seems to have taken a small hit to the delay.

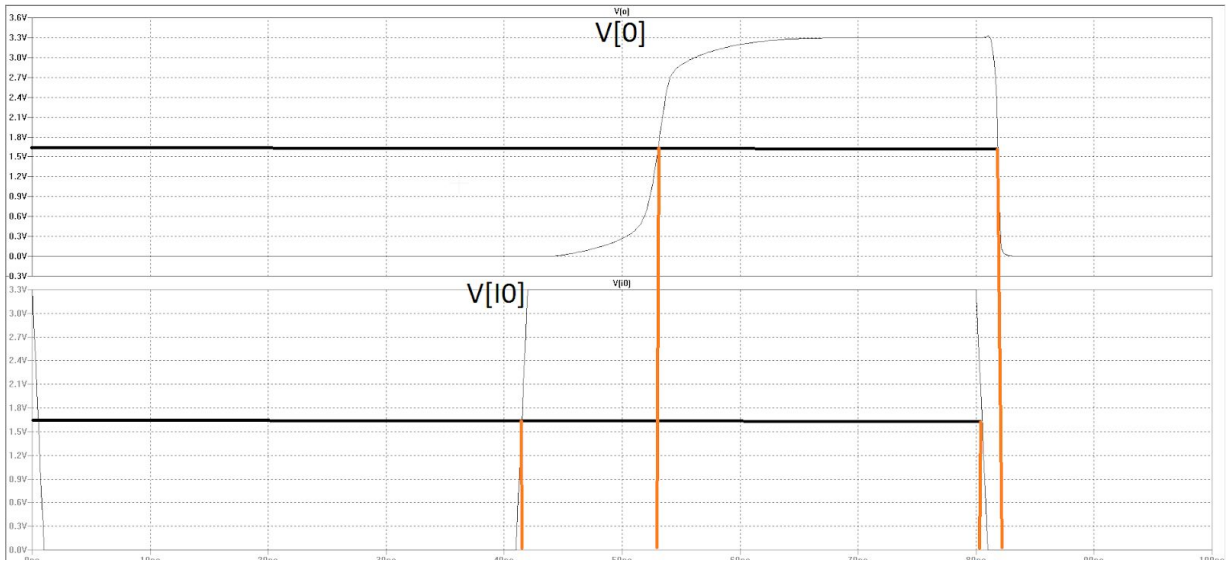


Figure 27. 32 Input AND Layout

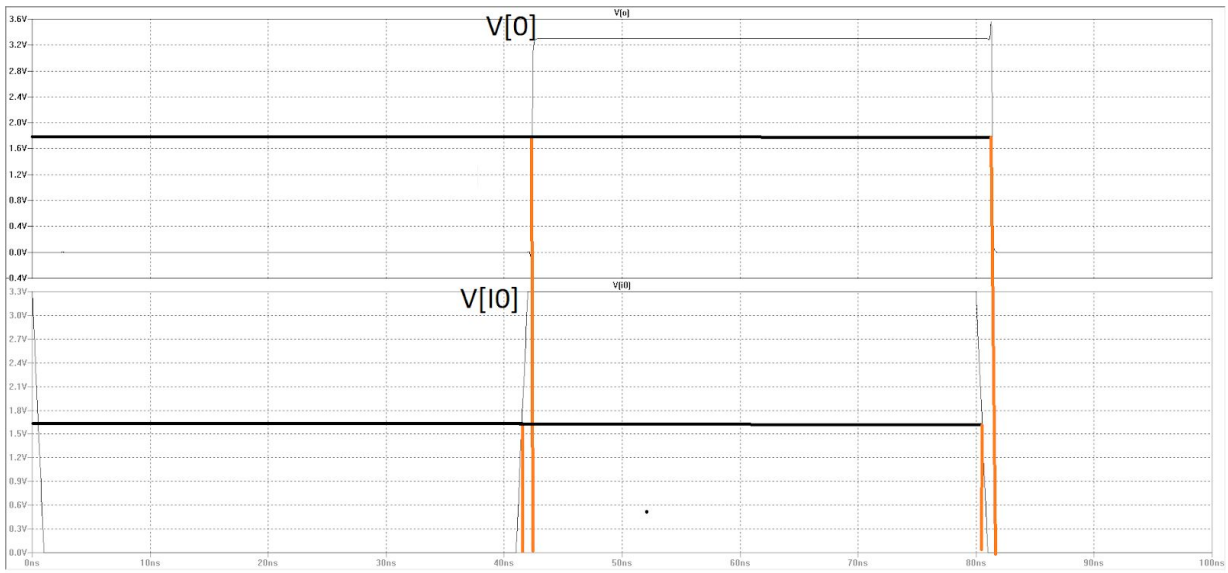


Figure 28. 32 Input AND Schematic

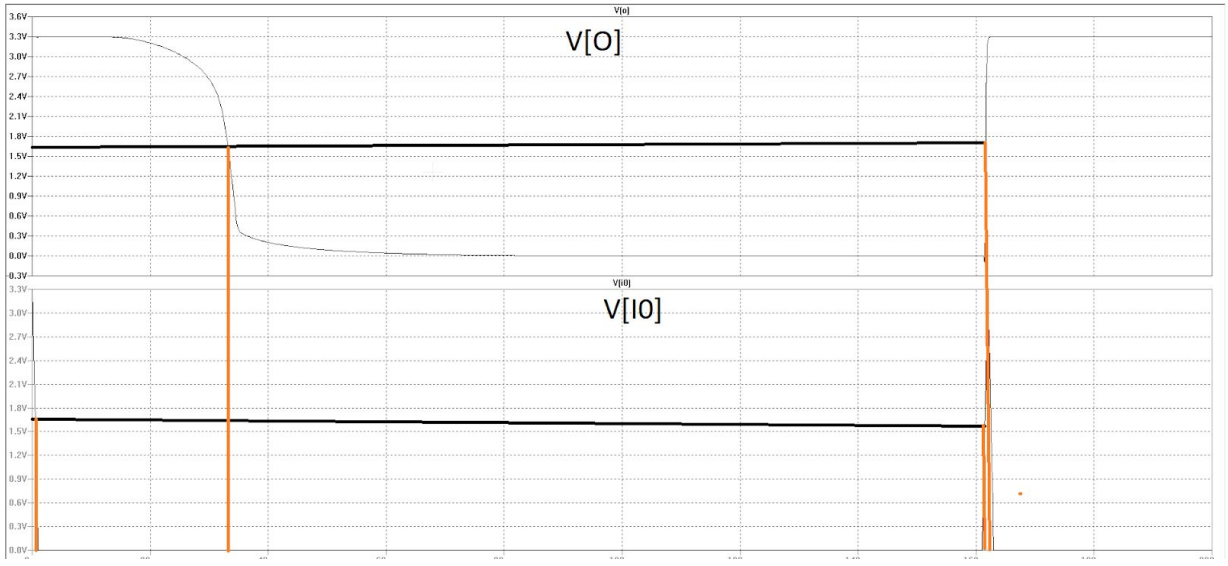


Figure 29. 32 Input OR Layout

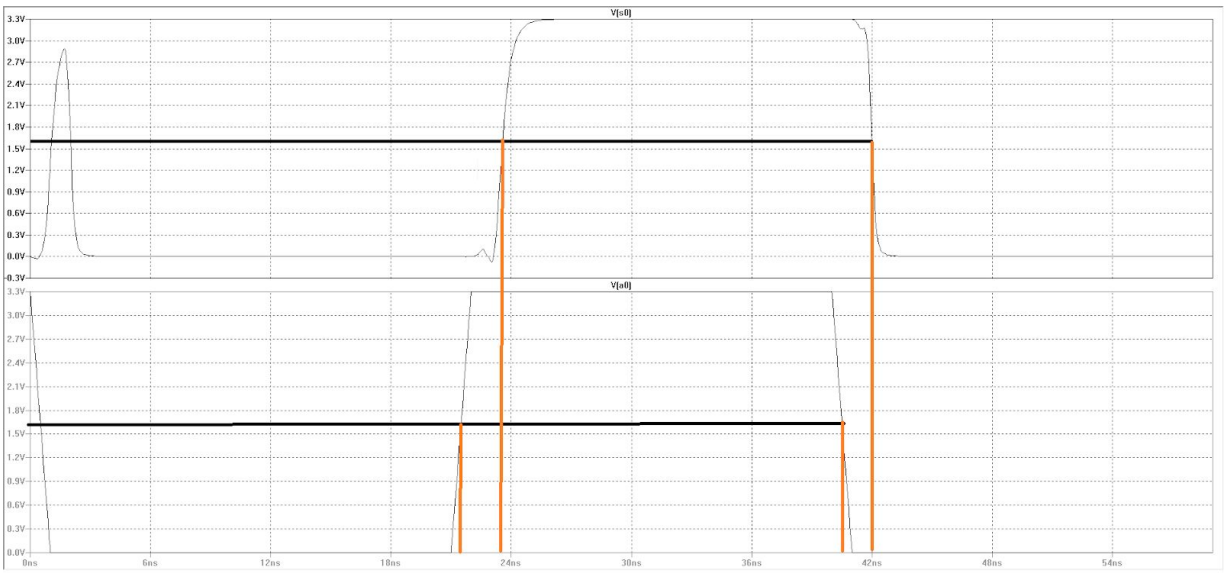


Figure 30. 32 Adders/Subtractor Layout

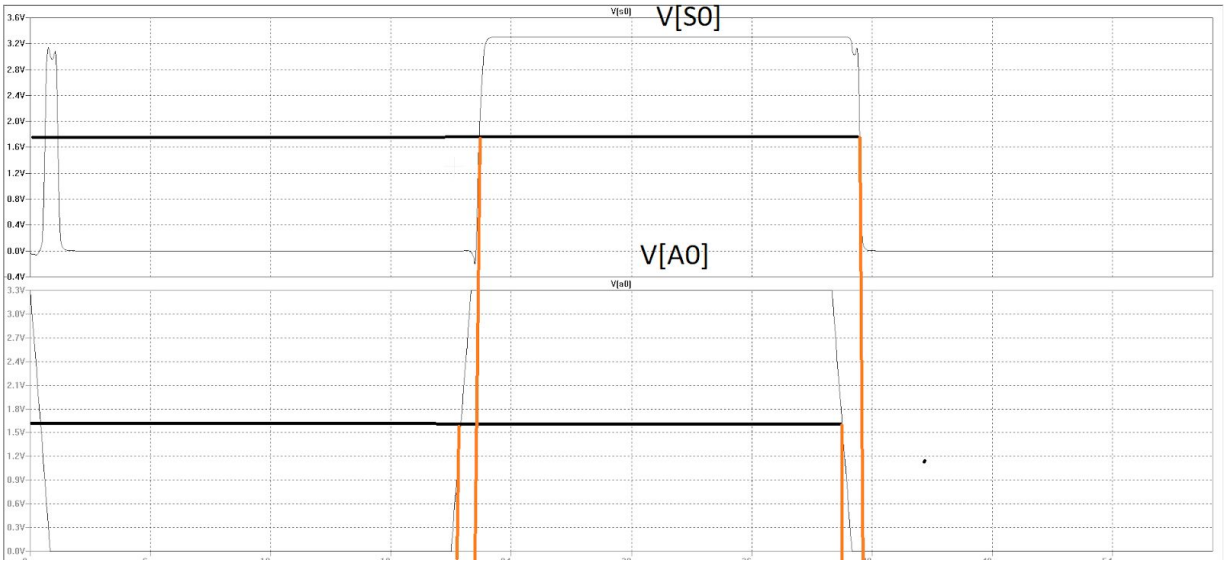


Figure 31. 32 Adders/Subtractor Schematic

When it came to testing the Carry Bit 31 it worked well on the Schematic however the layout seems to always stay on high, after testing it a bit a pattern arose where the a disturbance in the line shown in the box of Figure was the related to the input, the only thing I can assume is that the parasitics from all of the 32 AND and 1 OR gate that compose the Carry Bit 31 keep the component at always high as we've seen the 32 Input OR Gate requires very little to turn on thus some voltage from any of the AND could lead the the Carry staying permanently on.

Figure 32. Carry Bit 31 Layout

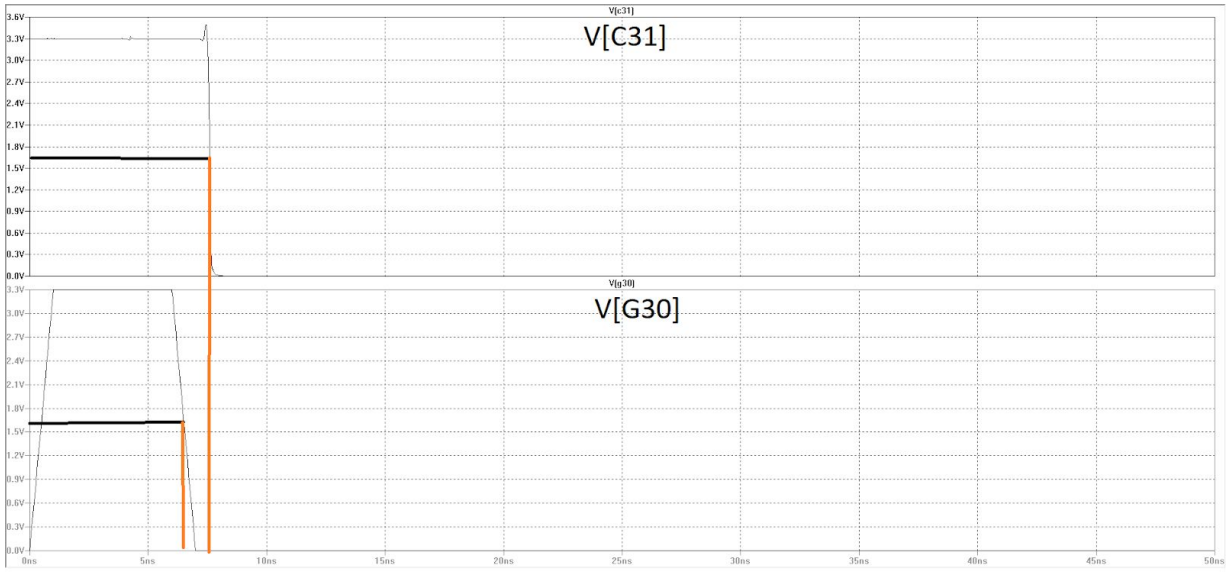


Figure 33. Bit 31 Layout

Table 6. Delay, Fall and Rise times of Components

Component	Schematic Delay	Schematic Fall/Rise	Layout Delay	Layout Fall/Rise
2 AND	0.204891*ns*	0.098913*ns* / .105*ns*	0.482323*ns*	0.163044*ns*/0.298913*n*

32 AND	0.9078865 ns	0.097826ns/0.089780 0ns	7.0724315ns	0.553341ns/4.673913 ns
2 OR	0.3066835 ns	0.10345ns/0.090717 ns	0.67391315n s	0.282608ns7/0.326087ns
32 OR	0.755698n s	0.1195652ns/0.060681ns	16.4203592n s	8.714597ns/0.28191n s
Carry Bit 1	0.7869873 ns	0.100735ns/0.092966ns	1.36956535n s	0.2826086ns/0.31283 ns
Carry Bit 31	0.750591	0.130435ns/0.072474ns	Couldn't Measure	Couldn't Measure
Full Adder/Subtractor	1.0760868 ns	0.195652ns/0.369652ns	1.97849445n s	0.494623ns/0.9247312ns
32 Bit Adder/Subtractor	0.502013n s	0.1716 ns/0.336384ns	1.480114ns	0.475807ns/0.888172 ns

It is clear from this that the AND and OR gates become very prone to parasitics as their delays skyrocket over the jump from schematic to layout. The Full Adder/Subtractor seems to work a lot better and is roughly two times slower than the schematic. 32 Bit Adder/Subtractor and Full Adder/Subtractor have same time because each adder in the 32 Bit Adder/Subtractor is built offf off indepent Full Adders and they are all provided with their inputs at the same time by the CLA, therefore it’s reasonable why they are close to each other. Unfortunately the Carry Bit 31 is impossible to test due to presumably the parasitics that come from all of the AND and OR gates that comprise it. Below are the LTspice codes used to acquire these images and data.

```
.global gnd vdd

*** TOP LEVEL CELL: 2_AND{sch}
Mnmos@2 net@41 In2 net@73 gnd N L=0.7U W=1.75U
Mnmos@3 net@73 In gnd gnd N L=0.7U W=1.75U
Mnmos@4 Out net@41 gnd gnd N L=0.7U W=1.75U
Mpmos@2 net@41 In2 vdd vdd P L=0.7U W=1.75U
Mpmos@3 net@41 In vdd vdd P L=0.7U W=1.75U
Mpmos@4 Out net@41 vdd vdd P L=0.7U W=1.75U

* Spice Code nodes in cell cell '2_AND{sch}'
VDD VDD 0 DC 3.3
VGND GND 0 DC 0
VIN In 0 PULSE(3.3 0 0 1n 1n 10n 20n)
VIN1 In2 0 PULSE(3.3 0 0 1n 1n 20n 40n)
.TRAN 0 50n
.include C:\electric\MOS_model.txt
.END
```

Figure 34.1 2 Input AND Schematic

```
* Spice Code nodes in cell cell '2_AND{lay}'
VDD VDD 0 DC 3.3
VGND GND 0 DC 0
VIN I0 0 PULSE(3.3 0 0 1n 1n 10n 20n)
VIN1 I1 0 PULSE(3.3 0 0 1n 1n 20n 40n)
.TRAN 0 40n
.include C:\electric\MOS_model.txt
.END
```

Figure 34.22 Input AND Layout

```
* Spice Code nodes in cell cell '32_And{sch}'
VDD VDD 0 DC 3.3
VGND GND 0 DC 0
VIN2 I0 0 PULSE(3.3 0 0 1n 1n 40n 80n)
VIN3 I1 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN4 I2 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN5 I3 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN6 I4 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN7 I5 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN8 I6 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN9 I7 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN10 I8 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN11 I9 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN12 I10 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN13 I11 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN14 I12 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN15 I13 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN16 I14 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN17 I15 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN18 I16 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN19 I17 0  PULSE(0 3.3 0 1n 1n 160n 320n)
VIN20 I18 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN21 I19 0 PULSE(0 3.3 0 1n 1n 160n 320n
VIN22 I20 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN23 I21 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN24 I22 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN25 I23 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN26 I24 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN27 I25 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN28 I26 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN29 I27 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN30 I28 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN31 I29 0  PULSE(0 3.3 0 1n 1n 160n 320n)
VIN32 I30 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN33 I31 0 PULSE(0 3.3 0 1n 1n 160n 320n)
.TRAN 0 100n
.include C:\electric\MOS_model.txt
.END
```

Figure 34.3 32 Input AND Schematic

```
* Spice Code nodes in cell cell '32_And{lay}'
VDD VDD 0 DC 3.3
VGND GND 0 DC 0
VIN2 I0 0 PULSE(3.3 0 0 1n 1n 40n 80n)
VIN3 I1 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN4 I2 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN5 I3 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN6 I4 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN7 I5 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN8 I6 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN9 I7 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN10 I8 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN11 I9 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN12 I10 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN13 I11 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN14 I12 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN15 I13 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN16 I14 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN17 I15 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN18 I16 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN19 I17 0  PULSE(0 3.3 0 1n 1n 160n 320n)
VIN20 I18 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN21 I19 0 PULSE(0 3.3 0 1n 1n 160n 320n
VIN22 I20 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN23 I21 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN24 I22 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN25 I23 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN26 I24 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN27 I25 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN28 I26 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN29 I27 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN30 I28 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN31 I29 0  PULSE(0 3.3 0 1n 1n 160n 320n)
VIN32 I30 0 PULSE(0 3.3 0 1n 1n 160n 320n)
VIN33 I31 0 PULSE(0 3.3 0 1n 1n 160n 320n)
.TRAN 0 100n
.include C:\electric\MOS_model.txt
.END
```

Figure 34.4 32 Input AND Layout

```
* Spice Code nodes in cell cell '2_Or{sch}'
VDD VDD 0 DC 3.3
VGND GND 0 DC 0
VIN In 0 PULSE(3.3 0 0 1n 1n 10n 20n)
VIN1 In2 0 PULSE(3.3 0 0 1n 1n 20n 40n)
.TRAN 0 40n
.include C:\electric\MOS_model.txt
.END
```

Figure 34.5 2 Input OR Schematic

```
* Spice Code nodes in cell cell '2_Or{lay}'
VDD VDD 0 DC 3.3
VGND GND 0 DC 0
VIN I0 0 PULSE(3.3 0 0 1n 1n 10n 20n)
VIN1 I1 0 PULSE(3.3 0 0 1n 1n 20n 40n)
.TRAN 0 40n
.include C:\electric\MOS_model.txt
.END
```

Figure 34.6 2 Input OR Layout

```
* Spice Code nodes in cell cell '32_Or{sch}'
VDD VDD 0 DC 3.3
VGND GND 0 DC 0
VIN2 I0 0 PULSE(3.3 0 0 1n 1n 20n 40n)
VIN3 I1 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN4 I2 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN5 I3 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN6 I4 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN7 I5 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN8 I6 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN9 I7 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN10 I8 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN11 I9 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN12 I10 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN13 I11 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN14 I12 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN15 I13 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN16 I14 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN17 I15 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN18 I16 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN19 I17 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN20 I18 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN21 I19 0 PULSE(3.3 0 0 1n 1n 160n 320n
VIN22 I20 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN23 I21 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN24 I22 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN25 I23 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN26 I24 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN27 I25 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN28 I26 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN29 I27 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN30 I28 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN31 I29 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN32 I30 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN33 I31 0 PULSE(3.3 0 0 1n 1n 160n 320n)
.TRAN 0 60ns
.include C:\electric\MOS_model.txt
.END
```

Figure 34.7 32 Input OR Schematic

```
* Spice Code nodes in cell cell '32_Or{lay}'
VDD VDD 0 DC 3.3
VGND GND 0 DC 0
VIN2 I0 0 PULSE(3.3 0 0 1n 1n 160n 50n)
VIN3 I1 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN4 I2 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN5 I3 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN6 I4 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN7 I5 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN8 I6 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN9 I7 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN10 I8 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN11 I9 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN12 I10 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN13 I11 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN14 I12 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN15 I13 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN16 I14 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN17 I15 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN18 I16 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN19 I17 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN20 I18 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN21 I19 0 PULSE(3.3 0 0 1n 1n 160n 320n
VIN22 I20 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN23 I21 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN24 I22 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN25 I23 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN26 I24 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN27 I25 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN28 I26 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN29 I27 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN30 I28 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN31 I29 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN32 I30 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN33 I31 0 PULSE(3.3 0 0 1n 1n 160n 320n)
.TRAN 0 200n
.include C:\electric\MOS_model.txt
.END
```

Figure 34.8 32 Input OR Layout

```
* Spice Code nodes in cell cell 'C1{sch}'
VDD VDD 0 DC 3.3
VGND GND 0 DC 0
VIN C0 0 PULSE(0 3.3 0 1n 1n 80n 160n)
VIN1 P0 0 PULSE(0 3.3 0 1n 1n 00n 160n)
VIN2 G0 0 PULSE(3.3 0 0 1n 1n 20n 40n)
.TRAN 0 40n
.include C:\electric\MOS_model.txt
.END
```

Figure 34.9 Carry Bit 1 Schematic

```
*** TOP LEVEL CELL: C1{lay}
X_2_AND@2 gnd P0 C0 net@64 vdd Project_3__2_AND
X_2_Or@1 gnd G0 net@64 C1 vdd Project_3__2_Or

* Spice Code nodes in cell cell 'C1{lay}'
VDD VDD 0 DC 3.3
VGND GND 0 DC 0
VIN C0 0 PULSE(0 3.3 0 1n 1n 80n 160n)
VIN1 P0 0 PULSE(0 3.3 0 1n 1n 00n 160n)
VIN2 G0 0 PULSE(3.3 0 0 1n 1n 20n 40n)
.TRAN 0 40n
.include C:\electric\MOS_model.txt
.END
```

Figure 34.10 Carry Bit 1 Layout

```
* Spice Code nodes in cell cell 'C31{sch}'
VDD VDD 0 DC 3.3
VGND GND 0 DC 0
VIN2 P0 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN3 P1 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN4 P2 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN5 P3 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN6 P4 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN7 P5 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN8 P6 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN9 P7 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN10 P8 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN11 P9 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN12 P10 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN13 P11 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN14 P12 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN15 P13 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN16 P14 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN17 P15 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN18 P16 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN19 P17 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN20 P18 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN21 P19 0 PULSE(3.3 0 0 1n 1n 160n 320n
VIN22 P20 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN23 P21 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN24 P22 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN25 P23 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN26 P24 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN27 P25 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN28 P26 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN29 P27 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN30 P28 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN31 P29 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN32 P30 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN33 P31 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN97 G0 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN34 G1 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN35 G2 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN36 G3 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN37 G4 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN38 G5 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN39 G6 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN40 G7 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN41 G8 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN42 G9 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN43 G10 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN44 G11 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN45 G12 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN46 G13 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN47 G14 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN48 G15 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN49 G16 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN50 G17 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN51 G18 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN52 G19 0 PULSE(3.3 0 0 1n 1n 160n 320n
```

```
* Spice Code nodes in cell cell 'C31{lay}'
VDD VDD 0 DC 3.3
VGND GND 0 DC 0
VIN2 P0 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN3 P1 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN4 P2 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN5 P3 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN6 P4 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN7 P5 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN8 P6 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN9 P7 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN10 P8 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN11 P9 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN12 P10 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN13 P11 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN14 P12 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN15 P13 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN16 P14 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN17 P15 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN18 P16 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN19 P17 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN20 P18 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN21 P19 0 PULSE(3.3 0 0 1n 1n 160n 320n
VIN22 P20 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN23 P21 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN24 P22 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN25 P23 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN26 P24 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN27 P25 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN28 P26 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN29 P27 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN30 P28 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN31 P29 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN32 P30 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN33 P31 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN97 G0 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN34 G1 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN35 G2 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN36 G3 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN37 G4 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN38 G5 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN39 G6 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN40 G7 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN41 G8 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN42 G9 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN43 G10 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN44 G11 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN45 G12 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN46 G13 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN47 G14 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN48 G15 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN49 G16 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN50 G17 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN51 G18 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN52 G19 0 PULSE(3.3 0 0 1n 1n 160n 320n
VIN53 G20 0 PULSE(3.3 0 0 1n 1n 160n 320n)
```

```
VIN53 G20 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN54 G21 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN55 G22 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN56 G23 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN57 G24 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN58 G25 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN59 G26 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN60 G27 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN61 G28 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN62 G29 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN63 G30 0 PULSE(0 3.3 0 1n 1n 20n 40n)
VIN65 C0 0 PULSE(3.3 0 0 1n 1n 160n 320n)
.TRAN 0 80n
.include C:\electric\MOS_model.txt
.END
```

Figure 34.11 Carry Bit 31 Schematic

```
VIN54 G21 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN55 G22 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN56 G23 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN57 G24 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN58 G25 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN59 G26 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN60 G27 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN61 G28 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN62 G29 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN63 G30 0 PULSE(0 3.3 0 1n 1n 20n 5n)
VIN65 C0 0 PULSE(3.3 0 0 1n 1n 160n 320n)
.TRAN 0 50n
.include C:\electric\MOS_model.txt
.END
```

Figure 34.12 Carry Bit 31 Layout

```
* Spice Code nodes in cell cell 'Full_Adder{sch}
VDD VDD 0 DC 3.3
VGND GND 0 DC 0
VIN A 0 PULSE(0 3.3 0 1n 1n 20n 40n)
VIN1 B 0 PULSE(3.3 0 0 1n 1n 80n 160n)
VIN2 C0 0 PULSE(3.3 0 0 1n 1n 80n 160n)
VIN3 C 0 PULSE(3.3 0 0 1n 1n 80n 160n)
.TRAN 0 40n
.include C:\electric\MOS_model.txt
.END
```

Figure 34.13 Full Adder Schematic

```
* Spice Code nodes in cell cell 'Full_Adder{lay}'
VDD VDD 0 DC 3.3
VGND GND 0 DC 0
VIN A 0 PULSE(0 3.3 0 1n 1n 20n 40n)
VIN1 B 0 PULSE(3.3 0 0 1n 1n 80n 160n)
VIN2 C0 0 PULSE(3.3 0 0 1n 1n 80n 160n)
VIN3 C 0 PULSE(3.3 0 0 1n 1n 80n 160n)
.TRAN 0 40n
.include C:\electric\MOS_model.txt
.END
```

Figure 34.14 Full Adder Layout

```
* Spice Code nodes in cell cell '32_Adder{sch}'
VDD VDD 0 DC 3.3
VGND GND 0 DC 0
VIN2 A0 0 PULSE(3.3 0 0 1n 1n 20n 40n)
VIN3 A1 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN4 A2 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN5 A3 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN6 A4 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN7 A5 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN8 A6 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN9 A7 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN10 A8 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN11 A9 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN12 A10 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN13 A11 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN14 A12 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN15 A13 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN16 A14 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN17 A15 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN18 A16 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN19 A17 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN20 A18 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN21 A19 0 PULSE(3.3 0 0 1n 1n 160n 320n
VIN22 A20 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN23 A21 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN24 A22 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN25 A23 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN26 A24 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN27 A25 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN28 A26 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN29 A27 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN30 A28 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN31 A29 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN32 A30 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN33 A31 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN97 B0 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN34 B1 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN35 B2 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN36 B3 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN37 B4 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN38 B5 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN39 B6 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN40 B7 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN41 B8 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN42 B9 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN43 B10 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN44 B11 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN45 B12 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN46 B13 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN47 B14 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN48 B15 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN49 B16 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN50 B17 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN51 B18 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN52 B19 0 PULSE(3.3 0 0 1n 1n 160n 320n
VIN53 B20 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN54 B21 0 PULSE(3.3 0 0 1n 1n 160n 320n)
```

```
* Spice Code nodes in cell cell '32_Adder{lay}'
VDD VDD 0 DC 3.3
VGND GND 0 DC 0
VIN2 A0 0 PULSE(3.3 0 0 1n 1n 20n 40n)
VIN3 A1 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN4 A2 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN5 A3 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN6 A4 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN7 A5 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN8 A6 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN9 A7 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN10 A8 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN11 A9 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN12 A10 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN13 A11 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN14 A12 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN15 A13 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN16 A14 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN17 A15 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN18 A16 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN19 A17 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN20 A18 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN21 A19 0 PULSE(3.3 0 0 1n 1n 160n 320n
VIN22 A20 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN23 A21 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN24 A22 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN25 A23 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN26 A24 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN27 A25 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN28 A26 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN29 A27 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN30 A28 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN31 A29 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN32 A30 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN33 A31 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN97 B0 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN34 B1 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN35 B2 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN36 B3 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN37 B4 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN38 B5 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN39 B6 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN40 B7 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN41 B8 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN42 B9 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN43 B10 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN44 B11 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN45 B12 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN46 B13 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN47 B14 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN48 B15 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN49 B16 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN50 B17 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN51 B18 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN52 B19 0 PULSE(3.3 0 0 1n 1n 160n 320n
VIN53 B20 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN54 B21 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN55 B22 0 PULSE(3.3 0 0 1n 1n 160n 320n)
```

```
VIN54 B21 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN55 B22 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN56 B23 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN57 B24 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN58 B25 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN59 B26 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN60 B27 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN61 B28 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN62 B29 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN63 B30 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN64 B31 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN65 C0 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN66 C1 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN67 C2 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN68 C3 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN69 C4 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN70 C5 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN71 C6 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN72 C7 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN73 C8 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN74 C9 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN75 C10 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN76 C11 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN77 C12 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN78 C13 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN79 C14 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN80 C15 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN81 C16 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN82 C17 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN83 C18 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN84 C19 0 PULSE(3.3 0 0 1n 1n 160n 320n
VIN85 C20 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN86 C21 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN87 C22 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN88 C23 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN89 C24 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN90 C25 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN91 C26 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN92 C27 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN93 C28 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN94 C29 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN95 C30 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN96 C31 0 PULSE(3.3 0 0 1n 1n 160n 320n)
.TRAN 0 59n
.include C:\electric\MOS_model.txt
.END
```

Figure 34.15 32 Adder Schematic

```
VIN56 B23 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN57 B24 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN58 B25 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN59 B26 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN60 B27 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN61 B28 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN62 B29 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN63 B30 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN64 B31 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN65 C0 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN66 C1 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN67 C2 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN68 C3 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN69 C4 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN70 C5 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN71 C6 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN72 C7 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN73 C8 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN74 C9 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN75 C10 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN76 C11 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN77 C12 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN78 C13 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN79 C14 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN80 C15 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN81 C16 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN82 C17 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN83 C18 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN84 C19 0 PULSE(3.3 0 0 1n 1n 160n 320n
VIN85 C20 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN86 C21 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN87 C22 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN88 C23 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN89 C24 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN90 C25 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN91 C26 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN92 C27 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN93 C28 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN94 C29 0  PULSE(3.3 0 0 1n 1n 160n 320n)
VIN95 C30 0 PULSE(3.3 0 0 1n 1n 160n 320n)
VIN96 C31 0 PULSE(3.3 0 0 1n 1n 160n 320n)
.TRAN 0 59n
.include C:\electric\MOS_model.txt
.END
```

Figure 34.16 32 Adder Layout

I have built the 32 Bit Adder/Subtractor Using CLA Logic in schematic view and even though DRC finds no errors whenever I try to use LTspice it take roughly five minutes to process the instruction followed by an error dealing with floating points or missing nodes which should not exist beauce DRC would have picked up on it. I cannot find why this happens, I must have connected something incorrectly on the lower level but it's impossible to pinpoint where.

Section 8: Measurements of Power, Delay, Chip Area, Timing, Number of Transistors for the Layout

As shown above in Table the timing seems to double on the well design units such as the lower level AND, OR gates and Carry Bits however once the number of transistors increase the parsticis take over and make the components unsable. The Adder/Subtractor on the other hand seems to only increase the delays by a slight bit which is to be expected as they all act individually making it much faster than if it were to work on ripple carry logic.

Below we see the image of the the size of the 32 Bit Adder/Subtractor Using CLA Logic layout the way to get the area in meters is 7720*350*10^-9*12753.5*250*10^-9 which is 0.00000861498 m or 8.61 um. As spoken about in the Layout section it's completely over size and could me much more compact.

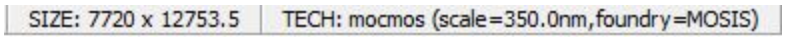


Figure 35 Layout Size

In this project I used

Table 7 Transistor Count

Component	Number of Components	Transistors
AND Gates	496	12,896
OR Gates	32	1,116
Full Adder/Subtractor	32	864
Total	560	14,876

The way I came up with these is as such every Carry signal has one 2 Input AND Gate which totals 31 2 Input AND Gates, each subsequent has been used one time less so 3 Input AND Gate occurs 30 times and so on. A 2 Input AND gate is consistent of 2 PMOS and 2 NMOS for the

NAND and 1 PMOS and 1 NMOS for the inverter totaling at 6, each subsequent one has two more transistors, below are my calculations.

AND Gates =
1*66+2*64+3*62+4*60+5*58+6*56+7*54+8*52+9*50+10*48+11*46+12*44+13*42+14*40+15*38+16*36+17*34+18*32+19*30+20*28+21*26+22*24+23*22+24*20+25*18+26*16+27*14+28*12+29*10+30*8+31*6

OR Gates =
66+64+62+60+58+56+54+52+50+48+46+44+42+40+38+36+34+32+30+28+26+24+22+20+18+16+14+12+10+8+6

Full Adder/Subtractor = 32*27

As mentioned in the layout section when it came to the choice of what kind of cmos design I was going to use I chose the basic setup we learned at the beginning of the class, transmission gates would make the creation of OR gate obsolete however that would also mean creating an inverter for every input possible which I believe would outweigh the amount of transistors used to create the OR gates. Secondly there is the choice of Dynamic gates which would in fact cut down the amount of transistors used in half plus two times number of components used, which is still a very significant number and would lead to much longer analysis during IRSIM and LTspice simulations so I've chosen not to use this method either.

Section 9: Conclusion and References

I've put around 60-70 hours into this project, I've created all the necessary components and the ultimate downfall came from inexperience from working on such a large scale, when I was building the components all of them seemed to work fine so I didn't bother checking them once they became overly large assume that they would work on the same principle. Obviously I was incorrect and that lead to a lot of the larger AND and OR gate being unstable to create the Carry Functions for the Carry Look Ahead logic. The only way to remedy that would be to rebuild it from scratch sadly I didn't have time to do that. This was a great lesson in the importance of checking the components throughout the designing process as well as a lesson on how to perform analysis of the delays, fall and rise times which helped me identify the issue by first coming up with speculations as to the issue with AND and OR gates and then verifying it with the parstice extraction. Overall I think I learned a decent about about working on a large project and how to make it work even though my project ultimately was unfinished.

References

Administrator. “Carry-Lookahead Adder.” *Electronics Hub*, 24 Dec. 2017, www.electronicshub.org/carry-look-ahead-adder/.