The City College of New York

Grove School of Engineering

EE 45700 – Digital Integrated Circuit

Spring 2017

Final Project: 32-Bit ALU

Instructor: Bruce Kim

May 22, 2017

Hasan Uchchas — Electrical Engineering

Azizur Rahman — Computer Engineering

Table of Content

I. Introduction

The invention of the transistor has pioneered the exponential growth of modern microelectronics. Various arrangements of these transistors have enabled us to perform Boolean calculations using machines. One of the most fundamental building blocks of any computing circuit, is an ALU. In this project, we designed a 32-bit ALU circuit using CMOS transistors.

ALU is an acronym for Arithmetic Logic Unit. An ALU is a combinational digital electronic circuit that performs arithmetic and bitwise operations on binary inputs. In this project, we were asked to design an ALU that performs four arithmetic operations, namely, addition, subtraction, multiplication, and division. We first built the four components required for the 32-bit ALU: a 32-bit Adder, a 32-bit Subtractor, a 32-bit Multiplier, and a 32-bit Divider. Then we connected each of the inputs to a 1:4 demultiplexer. In total, 64 demultiplexers were used for two 32-bit inputs. The control bits of the demultiplexers were connected. Thus, depending on the combination of the control bits, we were able to control which operation we want the ALU to perform on the inputs. All the schematics and layouts of the components were rigorously tested and simulated using both IRSIM and LTSpice to verify the functionality of the ALU.

At the block level, an ALU is represented with a trapezoidal symbol as depicted in Figure 1. Two signals are inputted and based on the control bits, the ALU performs the mathematical operation as instructed and outputs the result.

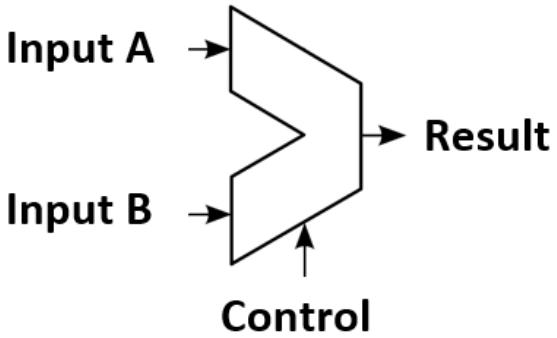


Figure 1. Block diagram of an ALU circuit.

The ALU that we have designed can perform four arithmetic operations such as addition, subtraction, multiplication, and division. Since we need to select between four types of operations, we needed two-bit control values. The truth table of the ALU is given below.

Table 1. Truth Table of the ALU.

Control A	Control B	Operation
0	0	Addition
0	1	Subtraction
1	0	Multiplication
1	1	Division

The 32-bit Adder circuit was designed using ripple-carry logic. We first designed a full-adder circuit using basic logic gates such as AND, OR, and XOR. We then cascaded 32 of these full-adder circuits to form the adder. The Carry-In of the first full-adder was connected to the ground since there is no carry-in when adding the first two bits.

The 32-bit subtractor was designed utilizing the Two's Complement property of binary numbers. In order to invert the bits of the negative number, we used an XOR gate in each of the inputs. We then connected one of the inputs of the XOR gate to the power,

thus essentially inputting a “1”. To add an extra “1” we connected the the first carry-in to the power as well.

Binary multiplication is very similar to decimal multiplication. In order to multiply the bits, we built a multiplier block using AND gates and full adders. Thus, by cascading 1024 of these blocks we built a 32-bit Multiplier circuit.

Similarly, the 32-bit Divider circuit was built by cascading 1024 divider blocks. Each divider block contains components such as inverter, full-subtractor, AND gates, and OR gates.

After building the schematics of all four components, we connected the inputs through 64 demultiplexers. The control bits of the demultiplexers were connected. Thus, by inputting the control signal as shown in table 1 we were able to select the operation to be performed by the ALU.

II. Circuit Schematics and Symbols

A. 32-Bit Adder

First we designed the schematic of the individual gates required to build a full adder. The gates required were AND gates, OR gates, and XOR gates. These gates were designed using CMOS logic. The AND gate was designed at first. Since CMOS logic implements an inverted output, an inverter was used at the output of the designed NAND gate to implement an AND gate. A gate level icon was developed for the AND gate. The designed circuit was tested and simulated in IRSIM to make sure that it is implementing the AND operation. The simulations matched with the expected outcomes.

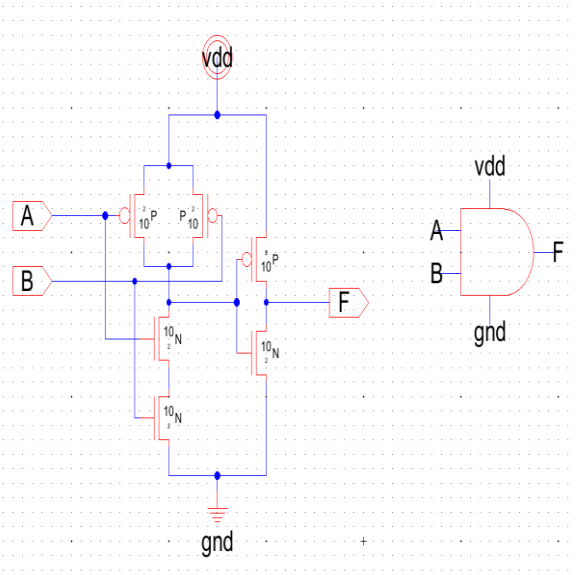


Figure 2. Schematic and icon of the AND gate.

Then the OR gate was designed using CMOS logic. Since CMOS design always implement the inverted function an inverter was used at the output of the designed NOR gate to implement the OR function. A gate level icon was designed for the OR gate. The designed circuit was tested and simulated in IRSIM to make sure that it is implementing the OR operation.

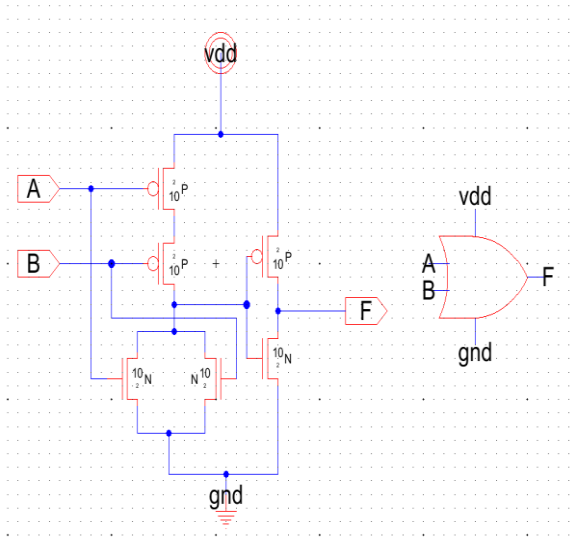


Figure 3. Schematic and icon of the OR gate.

Next the XOR gate was designed using CMOS logic. The Boolean expression of XOR logic is $\bar{A}B + \bar{B}A$. Since CMOS implements an inverted logic we needed to implement $(\bar{A}B + \bar{B}A)'$. Simplifying this expression, we get $AB + \bar{A}\bar{B}$. Therefore, by implementing $AB + \bar{A}\bar{B}$ using CMOS circuit we would implement the XOR function. A gate level icon was designed for the XOR gate. The designed circuit was tested and simulated in IRSIM to make sure that it is implementing the XOR operation.

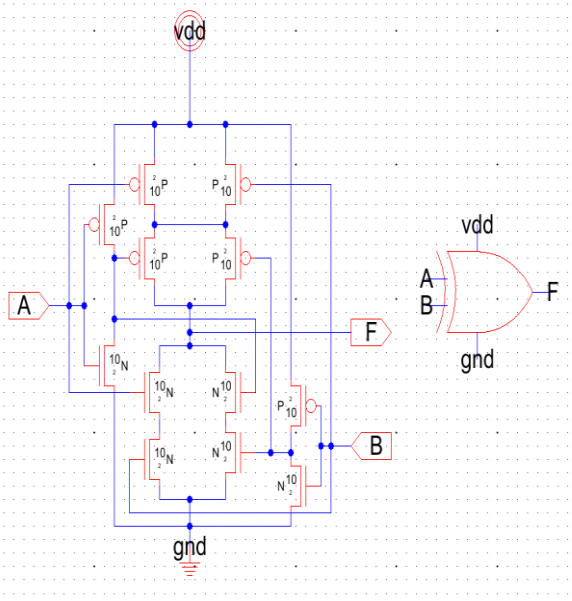


Figure 4. Schematic and icon of the XOR gate.

We can now use the designed icons to implement a 1-bit full adder circuit. Two XOR gates, two AND gates, and an OR gate were connected following the order discussed previously, to implement a 1-bit full adder circuit. All the gates were connected to the power supply (vdd) and the ground (gnd) nodes. An icon was created for the full adder. The designed circuit was tested and simulated in IRSIM to make sure that it is implementing the addition operation according to the truth table.

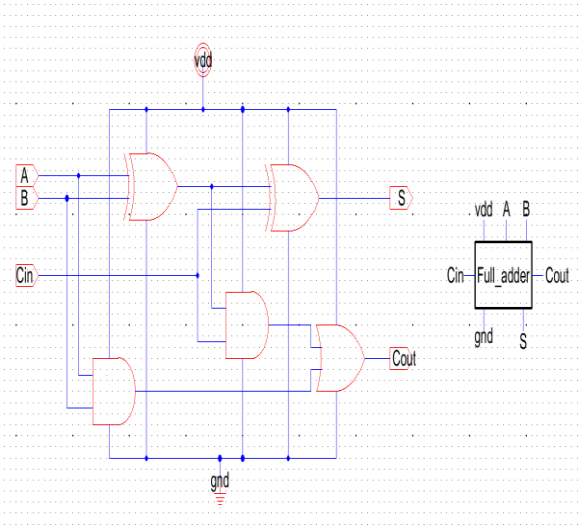


Figure 5. Schematic and icon of the full adder.

Table 2. Truth Table of the full adder.

A	B	Carry In	Carry Out	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

The designed full adder icon was then used to design a 32-bit adder. 32 full-adders were cascaded together to create the circuit.

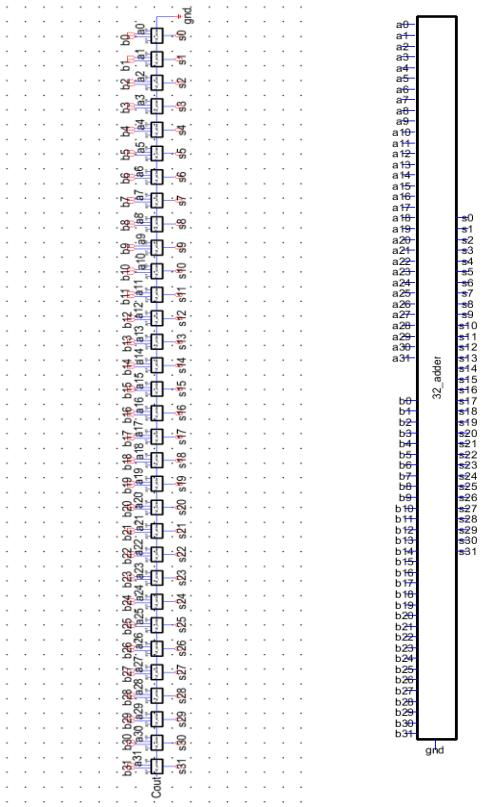


Figure 6. Schematic and icon of the 32-bit adder.

B. 32-Bit Subtractor

Using the Two's complement property of binary numbers, we can build a subtractor circuit from an adder circuit with minimal changes. According to the Two's complement property, the negative of a number is represented by inverting all the "0" bits to "1" and "1" bits to "0" and then adding a "1" to it.

To accomplish this task, we inputted the second number bits into XOR gates. According to the truth table of XOR gates, we can see that when one of the input is "1" then the output is the complement of the other input. When one input is "0" then the output is the same as the other input. For this subtractor circuit, we thus connected one inputs of each XOR gates to the vdd. To add the "1" required for the Two's complement property we connected the Carry-In bit of the first full adder to vdd as well. Thus, the input number will be converted to its Two's complement form and then added to the other input, essentially performing an addition between a positive number and a negative number. All the gates were connected to the power supply (vdd) and ground (gnd). The designed circuit was then tested and simulated in LTSpice and IRSIM. The outcomes of the simulations matched the expected outcomes.

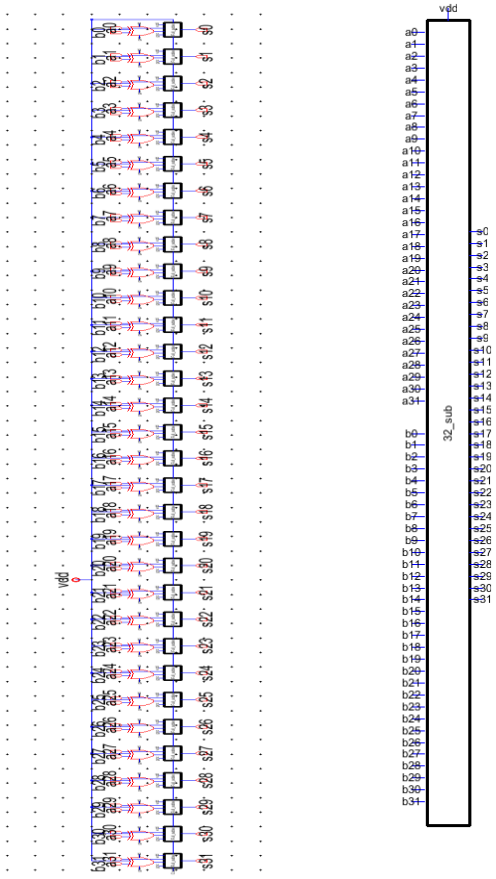


Figure 7. Schematic and icon of the 32-bit subtractor.

C. 32-Bit Multiplier

The designed full adder icon was used to design the building block of the multiplier circuit. The output of the AND gate was connected to one the of the inputs of the full adder. The inputs of the AND gate were extended to cascade the other blocks to form the multiplier circuit.

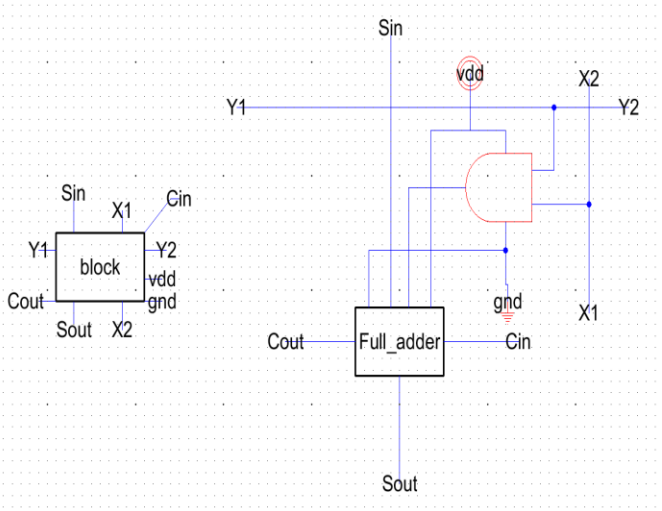


Figure 8. Building block of the multiplier circuit.

First we connected 32 blocks in a row. Then the next row of 32 blocks were built from the first row and the Carry-outs and Sum-outs were connected to the Carry-Ins and Sum-Ins. The process was repeated until we reached a column of 32 blocks. Thus, a 32 by 32 block circuit was formed.

Since the top row and the leftmost column does not have Carry-Ins and Sum-Ins, they were connected to the ground. All the output nodes were exported. The input nodes were named (a0-a31) for the first binary number and (b0-b31) for the second binary number. The output nodes were named (s0-s63). A total of 1024 blocks were used to build the circuit. The circuit was tested in IRSIM and LTSpice.

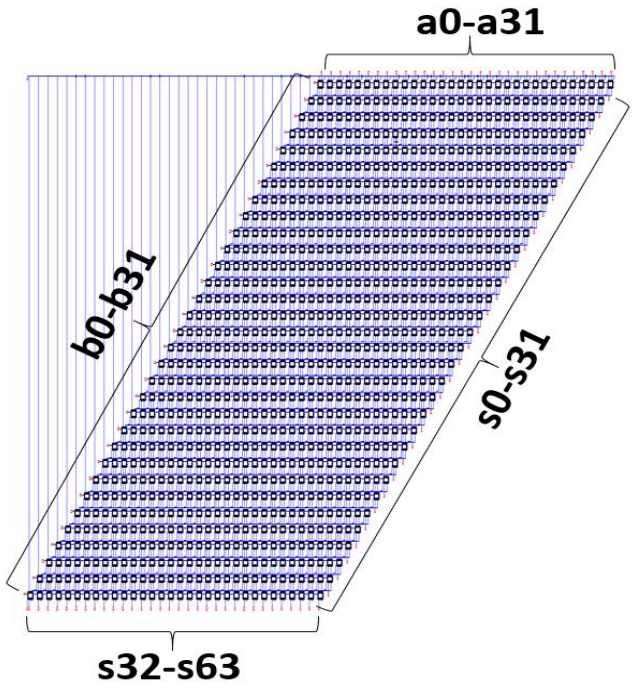


Figure 9. Schematic of the 32-bit multiplier.

D. 32-Bit Divider

To construct the divider circuit, we need a full subtractor circuit first. Using XOR, AND, OR and inverters we can easily build one. The schematic that we follow is given below:

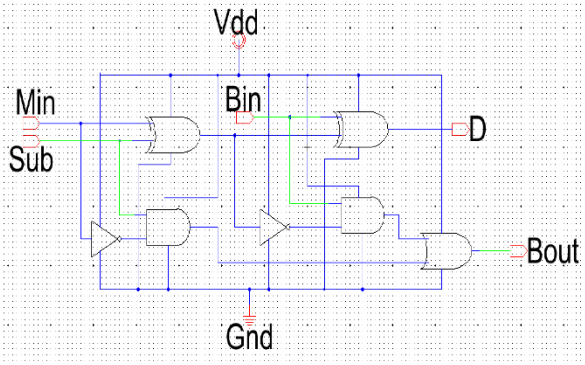


Figure 10. Schematic of full subtractor circuit.

Another circuit we need for Divider circuit is 2:1 Mux. The schematics is given below:

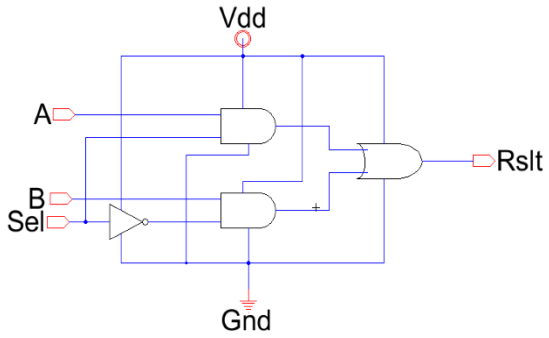


Figure 11. Schematic of 2:1 mux.

Div circuit is the modular block of 32-bit Divider circuit. It is simply built with a full subtractor and 2:1 MUX. It can be considered as 1-bit divider. 2 bits go in to the circuit and if one can be divided by another then it gives the quotient as result, otherwise just pass the input as result.

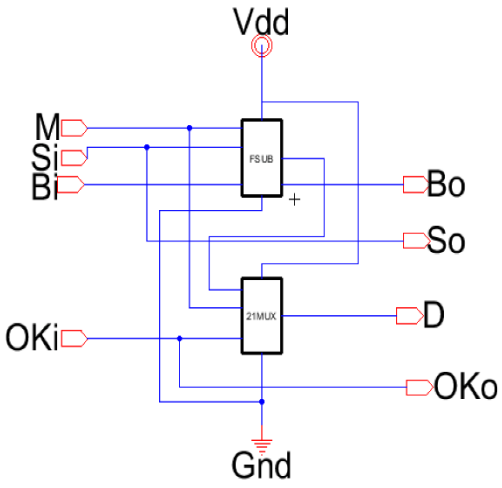


Figure 12. Schematic of the divider block.

The icon of the circuit which was used later is given below:

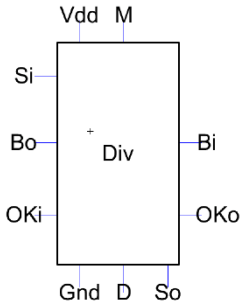


Figure 13. Icon of the divider block.

Using the Divider circuit, we built 32-bit divider. The theory behind the divider circuit is used from a previous class's schematics which shows how the Divider circuits should be connected. The connection between each div circuit is show below:

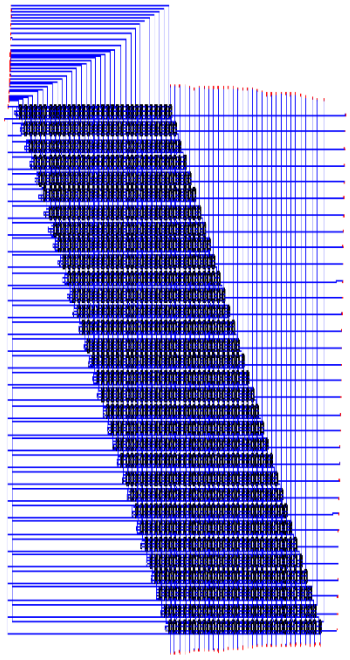


Figure 14. Schematic of the 32-bit divider.

E. 4:1 Demultiplexer

A demultiplexer (or demux) is a device which inputs multiple signals and then outputs a signal based on the control signals. For our 32-Bit ALU circuit, we need a 4:1 demux. The truth table of the demux is given below.

Table 3. Truth Table of 4:1 demux.

Input	A	B	Output 1	Output 2	Output 3	Output 4
D	0	0	D	0	0	0
D	0	1	0	D	0	0
D	1	0	0	0	D	0
D	1	1	0	0	0	D

To build the 4:1 demux we need four three-inputs AND gates and two inverters.

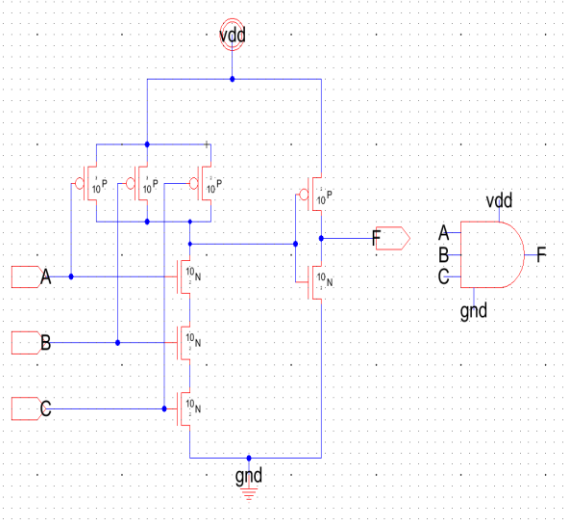


Figure 15. Three-input AND gate schematic and icon.

Using the designed three-input AND gates and inverters, we designed the 4:1 demux schematic. The circuit was tested in IRSIM and the outputs matched our truth table of the demux.

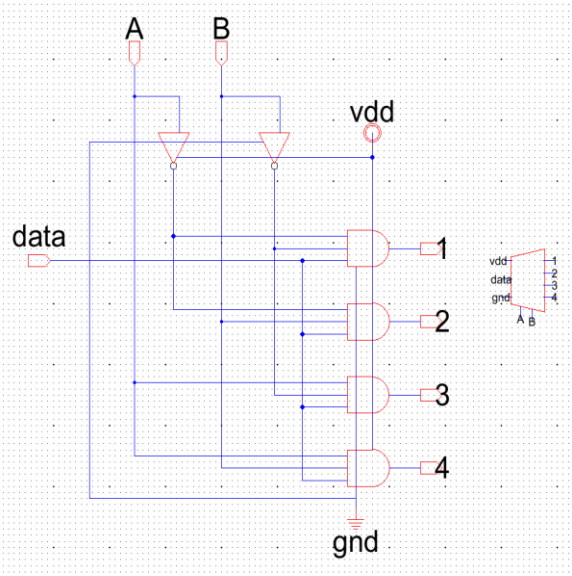


Figure 16. 4:1 Demux schematic and icon.

F. 32-Bit ALU

Each of the inputs (a0-a7) and (b0-b7) were connected to a 4:1 demux. The control signals of the demuxes were connected. In this formation when the control signal is (A, B) = (0,0), all the inputs are fed to the Adder. Similarly, when it is (0,1), (1,0), and (1,1) the inputs are fed to the Subtractor, Multiplier, and Divider respectively. The icons of the 32-Bit Adder, 32-Bit Subtractor, 32-Bit Multiplier, and 32-Bit Divider were used to build the 32-Bit ALU circuit.

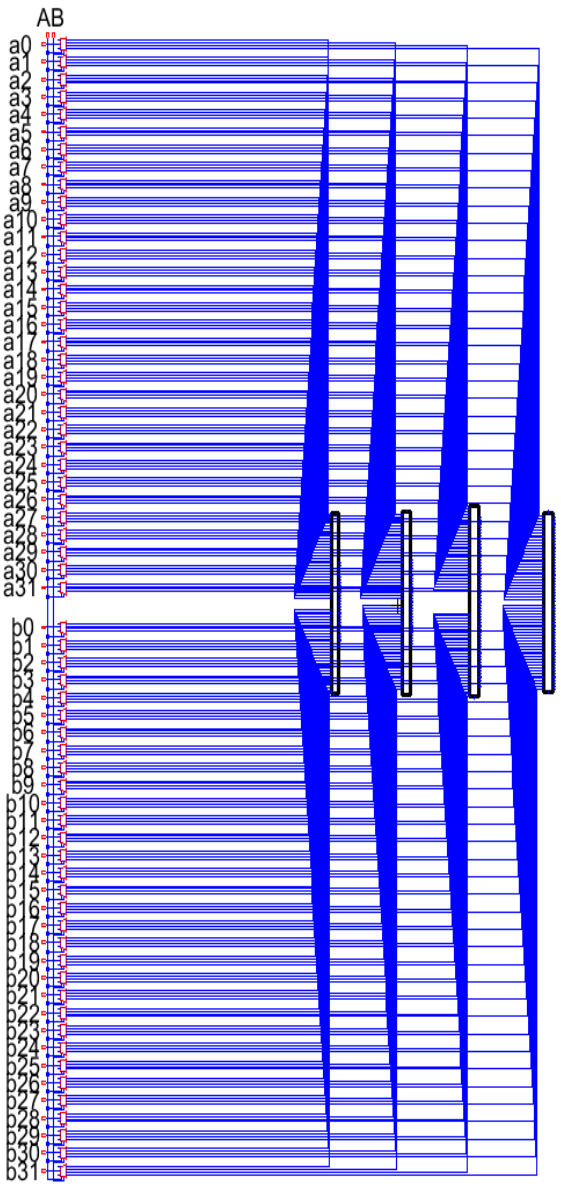


Figure 17. 32-Bit ALU schematic

III. Circuit Layouts

A. 32-Bit Adder

A stick diagram of the AND gate was designed following the Euler path A-B. Now the layout of the AND gate was designed following the stick diagram.

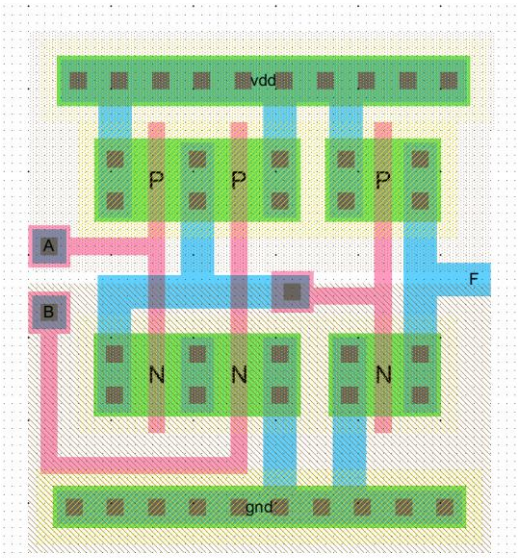


Figure 18. Layout of the AND gate.

The stick diagram of the OR gate was designed following the Euler path A-B. Now the layout of the OR gate was designed following the stick diagram.

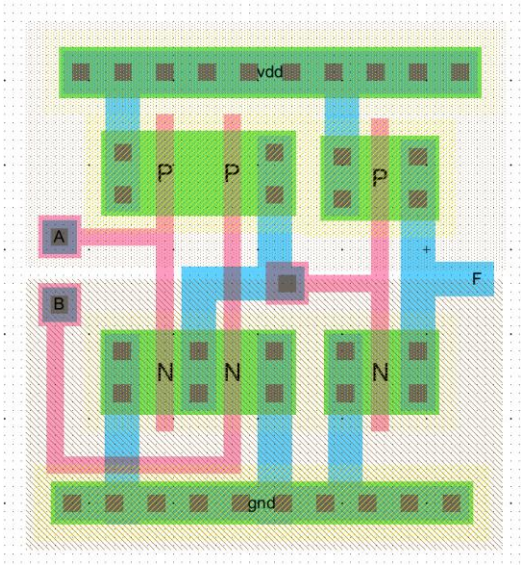


Figure 19. Layout of the OR gate.

The stick diagram of the XOR gate was designed following the Euler path A-B. Now the layout of the XOR gate was designed following the stick diagram.

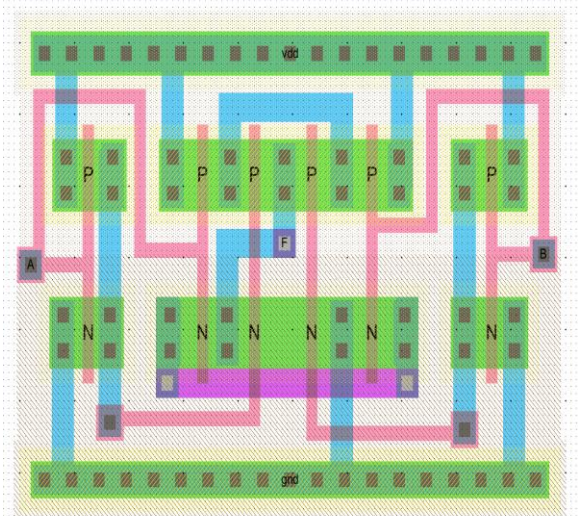


Figure 20. Layout of the XOR gate.

The full adder was designed by combining two XOR gates, two AND gates, and one OR gate following the full adder schematics designed in the previous sections. The gates were placed from left to right on a linear plane. Then the necessary connections were made with metal contacts and metal layers. Metal 1, metal 2, metal 3 layers were used when necessary to avoid any cross connections.

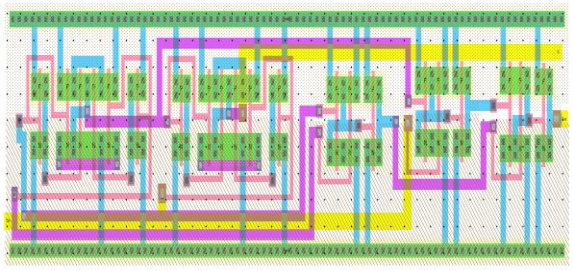


Figure 21. Layout of the full adder

The layout of the 8-bit ripple carry adder/subtractor were designed by connecting 8 full adders in a cascade formation. The C-out's of each full adders were connected to the C-in's of the next one. The C-in of the first adder was used as the "Select" input just like the schematics. The layout of the adder/subtractor is given below:

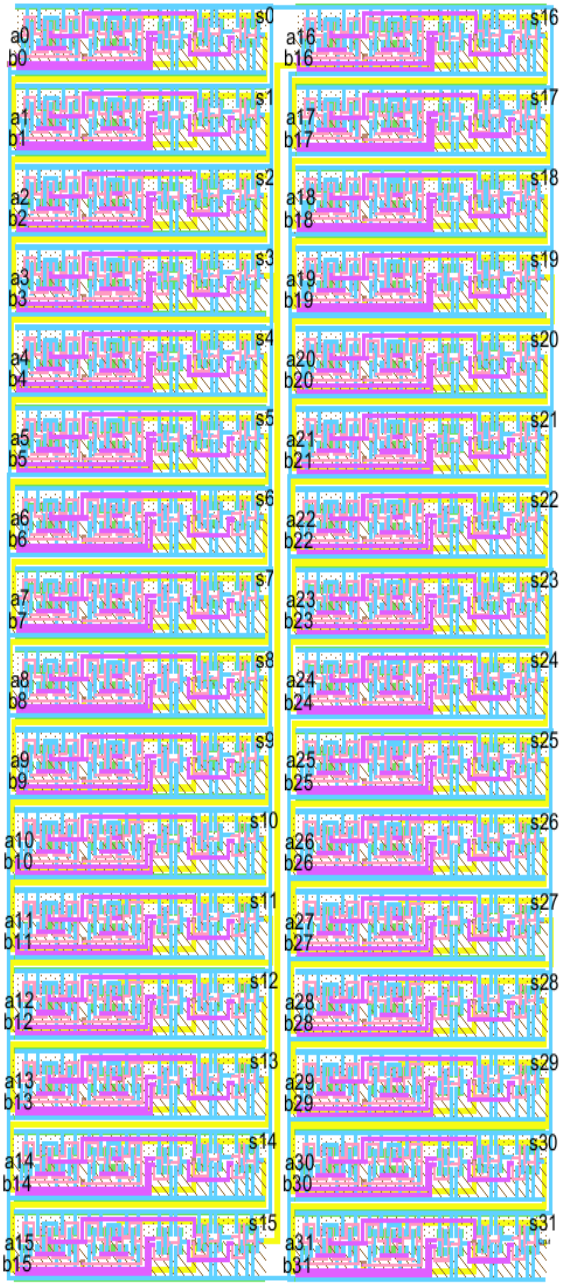


Figure 22. Layout of the 32-bit adder.

B. 32-Bit Subtractor

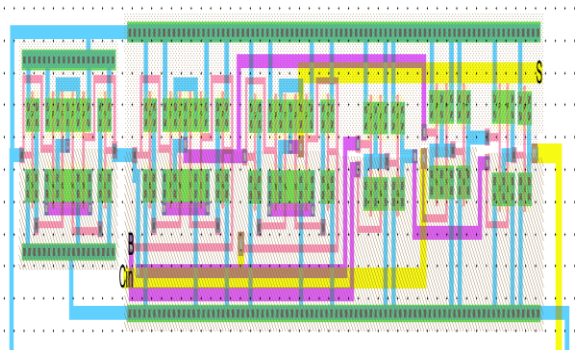


Figure 23. Layout of the subtractor block.

The Full Adder layout was connected to an XOR gate layout to form the building block of the subtractor circuit.

Figure 24. Layout of 32-bit subtractor.

C. 32-Bit Multiplier

The block was designed in such a way that it represents the schematic block that was designed in the schematic section.

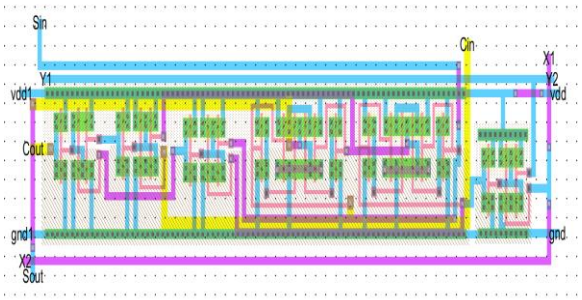


Figure 25. Layout of the multiplier block.

By cascading 1024 of these blocks the multiplier circuit layout was built.

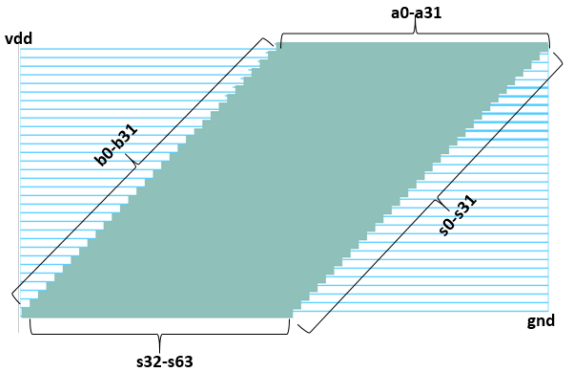


Figure 26. Layout of the 32-bit multiplier.

Since the circuit was too large Electric could not generate the details of the circuit when zoomed out.

Figure 27. Zoomed-in view of the layout.

This is a zoomed in view of the mid-section of the layout. The cascading method of the layout was the same as the schematic. We connected the Carry-ins and Sum-Ins to the Carry-outs and Sum-outs of the next stage blocks.

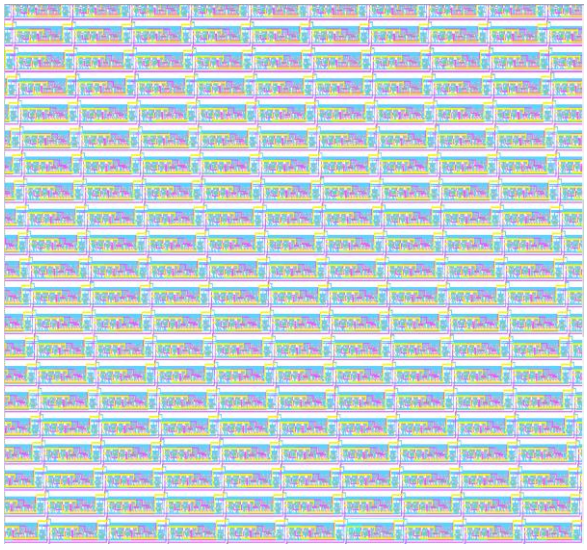


Figure 28. Zoomed-in view of the layout.

D. 32-Bit Divider

Similar to the multiplier layout, the 32-bit Divider layout was built by cascading 1024 divider blocks.

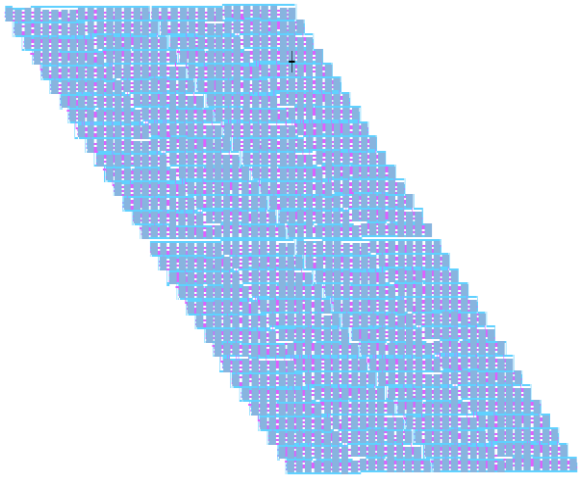


Figure 29. Layout of the 32-bit divider.

E. 4:1 Demultiplexer

The demux layout was designed using the three-input AND gates and inverters. The layout was tested and it matched with the 4:1 demux truth table. DRC, ERC, and NCC was done and no error was found.

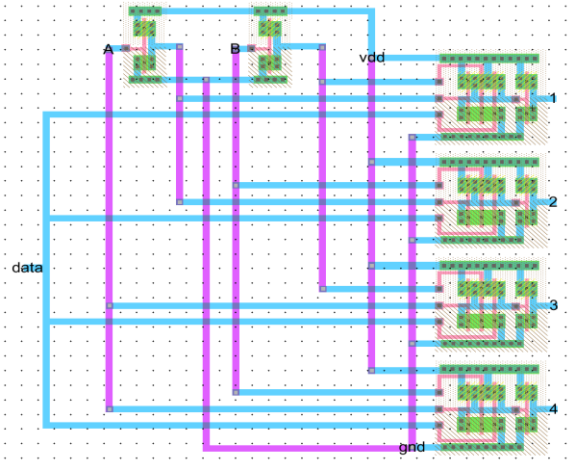
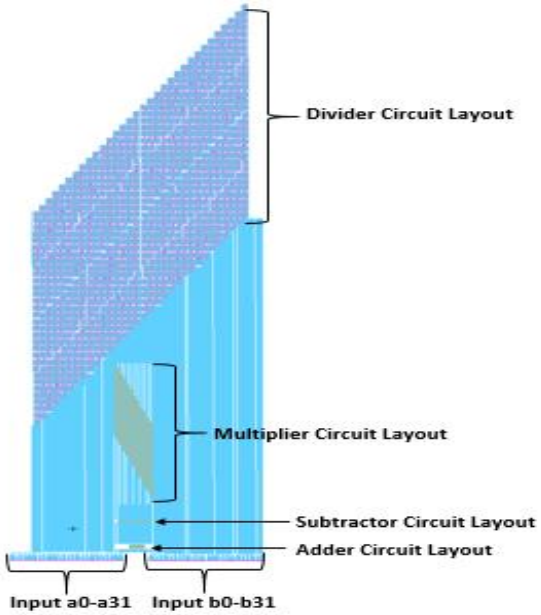


Figure 30. Layout of the 4:1 demux.

F. 32-Bit ALU

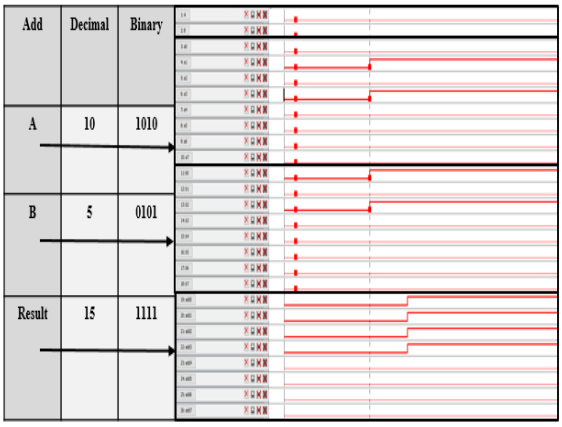
The 32 Bit ALU layout was built by bringing the layouts of the 32-bit adder, 32-bit subtractor, 32-bit multiplier, and the 32-bit divider layouts into the same cell.

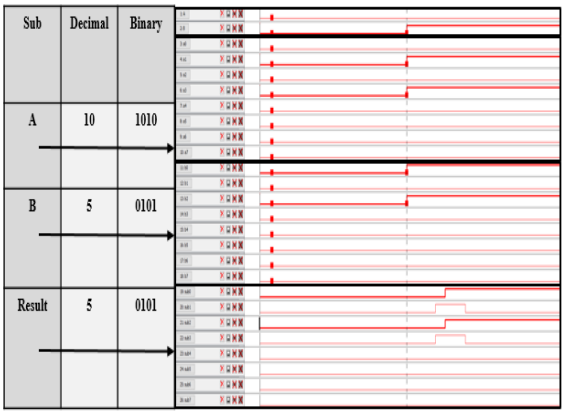


IV. Simulation Results

A. IRSIM

We simulated the ALU circuit in IRSIM to test its functionality. We inputted two binary values 1010 and 0101 which translates to 10 and 5 in decimal respectively, and we observed the output when we change the control bits. The results are shown below:





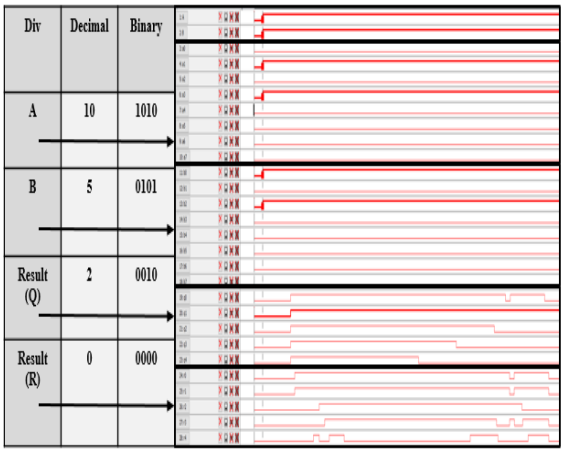


Figure 31. IRSIM simulations

The simulation clearly shows the expected outcomes of the operations depending on the control bits. Therefore, our ALU circuit is operating accordingly.

B. LTSPICE

The ALU circuit was simulated in LTSPICE to measure the rise time, fall time, propagation delay and the total power consumption.

Figure 32. Rise time of the ALU.

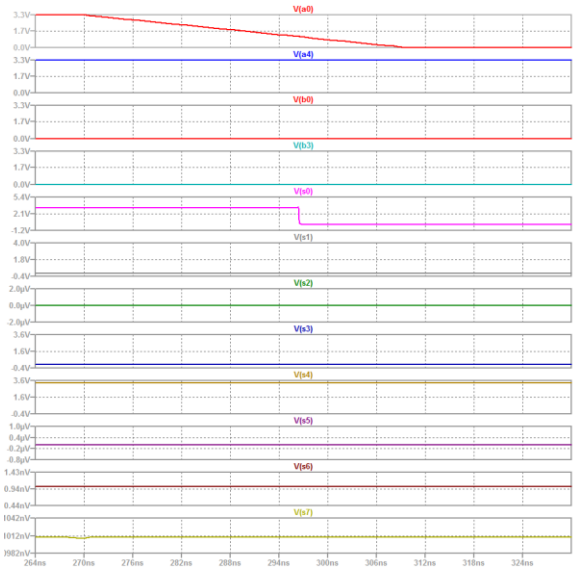


Figure 33. Fall time of the ALU.

Rise Time = 2 ns

Fall Time = 2.5 ns

$t_{pHL} = 7$ ns

$t_{pLH} = 4\ ns$

$$Propagation\ delay, t_p = \frac{7\ ns + 4\ ns}{2}$$

$$= 5.5\ ns$$

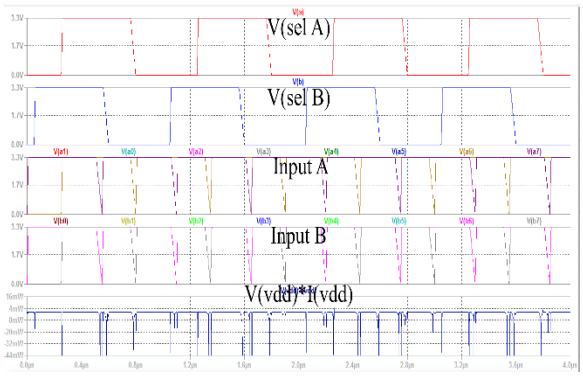


Figure 34. Total power dissipation

Total power dissipation was found to be around 27 mW.

V. Conclusion

The goal of the project was to design a 32-bit ALU that can perform addition, subtraction, multiplication, and division. In order to do so, we first designed the individual components such as 32-bit adder, 32-bit subtractor, 32-bit multiplier, and 32-bit divider circuits. We simulated these circuits individually to make sure that the designed circuits are working perfectly. After testing all the components, we connected the entire 32-bit ALU. We then tested the ALU in IRSIM and LTSpice to make sure that it is operating properly. Overall, the project was a success and it helped us appreciate the intricate details of digital circuit design.

VI. References

[1] “Arithmetic Logic Unit (ALU) from Ati.ttu.ee. [Online]. Available: http://ati.ttu.ee/IAY0340/labs/Tutorials/SystemC/ALU.html. Accessed: May 10, 2017.

[2] "Electric VLSI tutorials from CMOSedu.com," in CMOSedu.com. [Online]. Available: http://cmosedu.com/videos/electric/electric_videos.htm. Accessed: Feb. 26, 2017.

[3] J. M. Rabaey, A. P. Chandrakasan, B. Nikolić, and B. Nikolic, Digital integrated circuits: A design perspective, 3rd ed. United States: Pearson Education, 2012.