Digital Design Lab

ECE 315

Lab/Project #3

4 Bit Adder & 4 Bit Subtractor

Group #7

Karysse Hay, Nikeem

Dunkelly-Allen and Dibyanshi Mishra

Zeinab Ramezani, TA

University of Miami

2/28/2022

Overview

This lab required us to design and implement a one-bit adder that would then be combined with 3 others like itself to create a 4 one-bit adder on Quartus Prime. The 4 one-bit adder adds the bits X and Y and the carry from the previous column called the carry in (cin) and produces the outputs the sum bit (sum) and the carry out bit (cout). The ‘sum’ bit gives the value of the least significant bit of the sum and ‘cout’ gives the out carry. The final adder is capable of adding two numbers (0-15).

The next part was to modify the 4 one-bit adder to create a 4 Bit Adder/Subtractor. In Digital Circuits, A Binary Adder-Subtractor is one which is capable of both addition and subtraction of binary numbers in one circuit itself. This was done by using four multiplexers and in cooperating the idea that a negative binary number can be represented using 2’s complement.

Equipment

- Quartus Prime Software
- AND gates
- OR gates
- XOR gates
- NOT gates
- ModelSim
- Input Pins
- Output Pins
- Orthogonal Node
- Diagonal Node

Description

Firstly, the initial truth table for a one-bit adder (Table 2) was completed using logic and the binary addition truth table (Table 1). There are three inputs: X and Y represent the 2 decimal numbers being added, and ‘Cin’ represents the carry overflow digit from the previous addition of 2 inputs. Two K-Maps were derived from the truth table for the two outputs: ‘Sum’ and ‘Cout’. Sum represents the addition of the 2 inputs, whereas Cout represents the scenario where an extra bit is needed to display the sum of the two inputs X and Y. The high outputs were grouped, and two Boolean equations were found. These consisted of: X’Y’Cin + X’YCin’ + XY’Cin’ + XYCin and XY + YCin + XCin. These were simplified further to the equations: $X \oplus Y \oplus Cin$ and $XY+Cin(X \oplus Y)$ respectively.

After the logic was completed, a project using the Quartus Prime software was created. After the project was launched, New Project Wizard was selected. The location and name of the file was chosen, an empty project was selected, and the flowing wizard was left blank. For the next selection, the DE1 SoC board was chosen. Under simulation the Tool Name was changed to simulations and the format to VHDL. After the window changed from the file menu, a new Block Diagram/Schematic option was chosen.

Once the set up was done, the circuit for a one-bit adder was created using two AND gates, two XOR gates and one OR gate. It consisted of three inputs: X, Y and Cin, and two outputs Sum and Carry, as mentioned above. The circuit was compiled and simulated. Various combinations of inputs were tested to ensure the final product was working correctly. A Symbol File was then created to represent the one-bit adder.

In another Block/Schematic file, four one-bit adders were added in series to give five outputs: Q_0, Q_1, Q_2 and Q_3 and a single Cout bit with Cout being the most significant. However, in the second adder, instead of an entirely new Cin input, the Cout from the previous adder was connected. Therefore, the Cout from each adder besides the last one became the Cin for the next adder in series. This can be better seen in the Logic Diagram. This new design was compiled and simulated and tested to ensure it was working completely as well.

Finally, after the full adder was functioning completely, a new symbol file was created to illustrate the 4 one-bit adder. This was connected to four multiplexers to implement the 4-bit

Adder/ Subtractor. This consisted of the 4 one-bit adder, four multiplexers, five NOT gates and seven inputs: A_0, A_1, A_2, A_3, B_0, B_1, B_2, B_3 and Cin and five outputs: Q_0, Q_1, Q_2, Q_3 and Cout (Sign Bit). The circuit was compiled and simulated/ tested for a final time.

Design Synthesis

1. TRUTH TABLES

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 1: Binary Addition Truth Table

Table 1 was used in order to fill out the missing values for Table 2 (One-Bit Full Adder).

X	Y	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 2: One-Bit Full Adder Truth Table

K-Maps were constructed using the three inputs (X, Y and Cin) for each of the outputs. The first K-Map was done using the column for the Sum bit and the second one used the Cout column.

2. OUTPUT K-MAPS

Figure 1: K-Map for Sum

Figure 2: K-Map for Cout

3. BOOLEAN EQUATIONS

Equation for Figure 1:

a. X'Y'Cin + X'YCin' + XY'Cin' + XYCin

b. X ⊕Y⊕Cin (Simplified/ Reduced Version)

Equation for Figure 2:

a. XY + YCin + XCin

b. XY+Cin(X⊕Y) (Simplified/ Reduced Version)

Complete Logic Diagram

1. One-Bit Full Adder

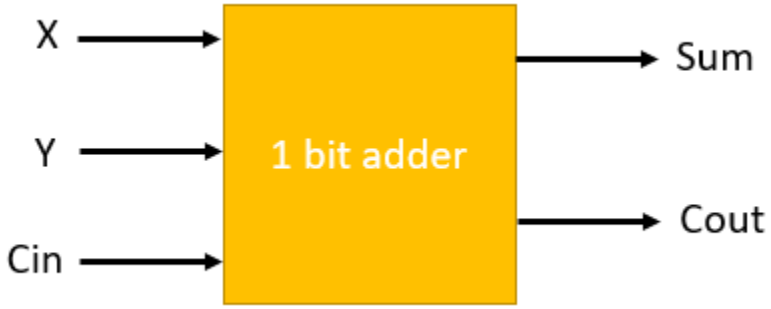

Figure 3: Block Diagram for One-Bit Adder

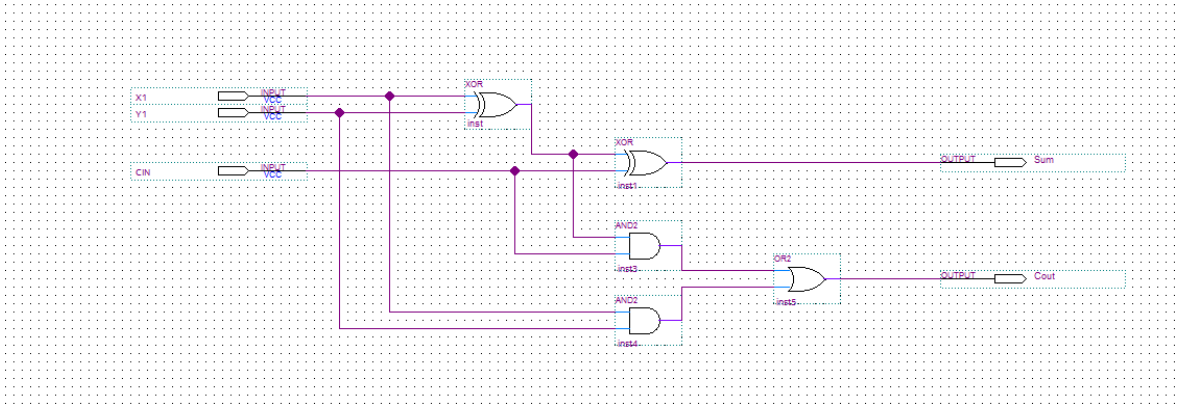

Figure 4: Complete Logic Diagram for One-Bit Adder on Quartus Prime

2. 4 One-Bit Full Adder

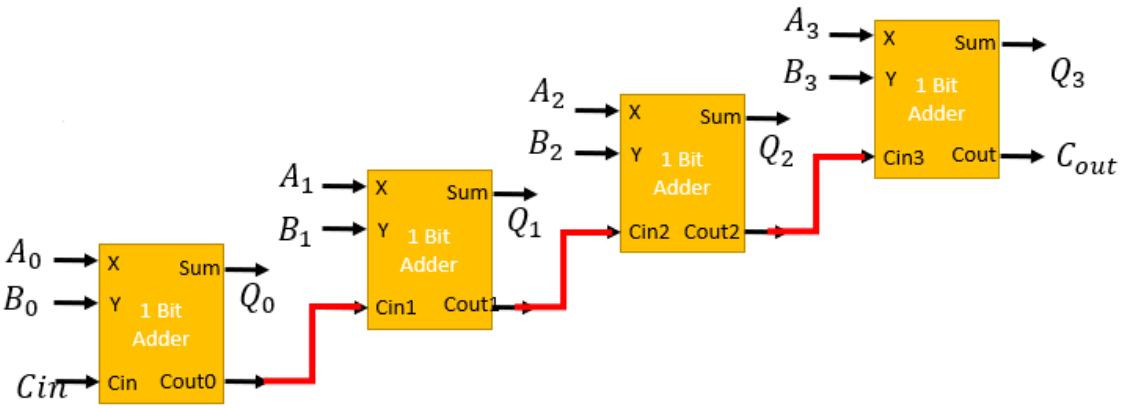

Figure 5: Block Diagram for the 4 One-Bit Full Adder

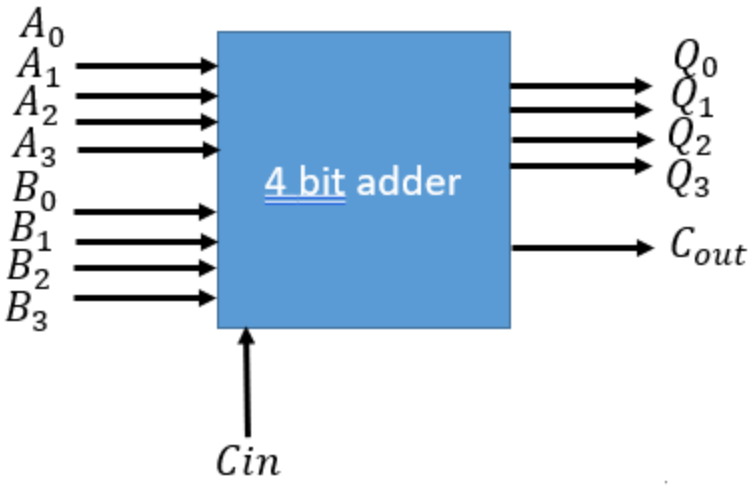

Figure 6: Block Diagram for 4-Bit Full Adder (Simplified)

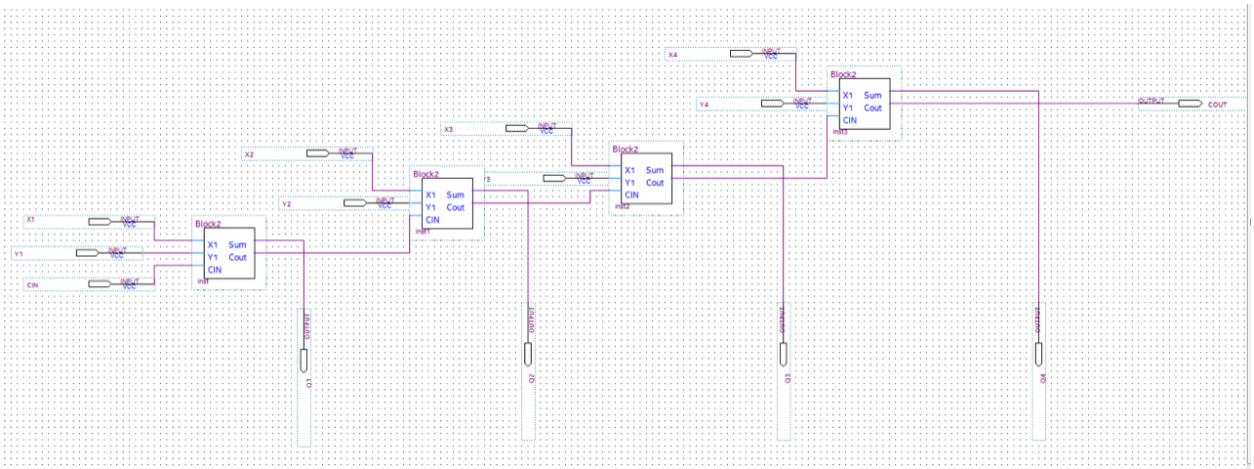

Figure 7: Complete Logic Diagram for 4-Bit Full Adder on Quartus Prime

3. 4 One-Bit Subtractor

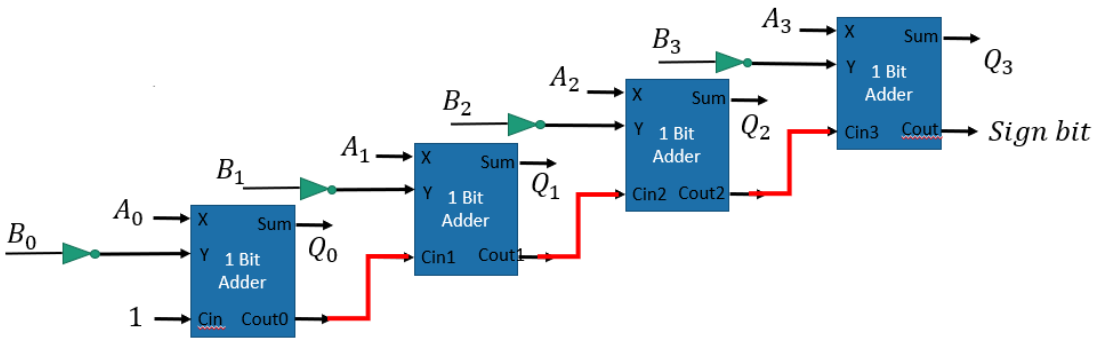

Figure 8: Block Diagram for the 4 One-Bit Full Subtractor

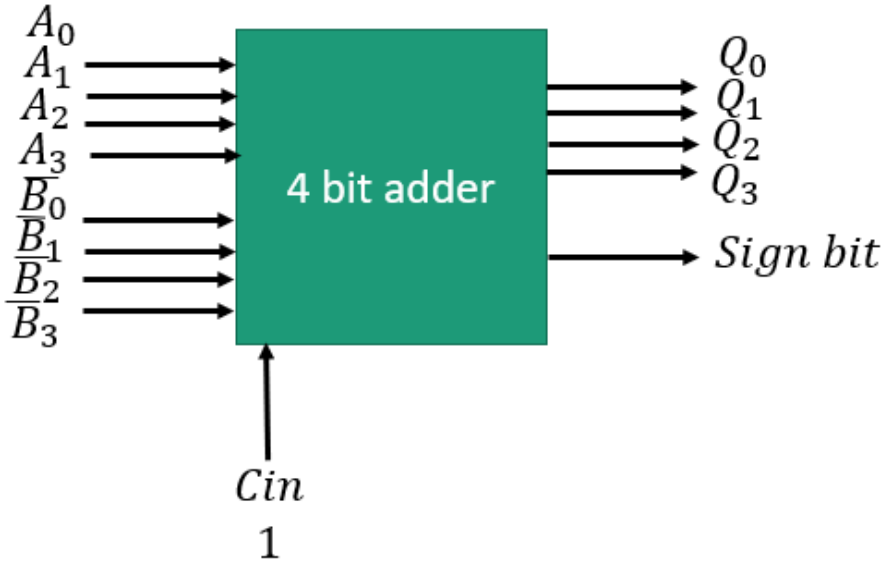

Figure 9: Block Diagram for 4-Bit Full Subtractor (Simplified)

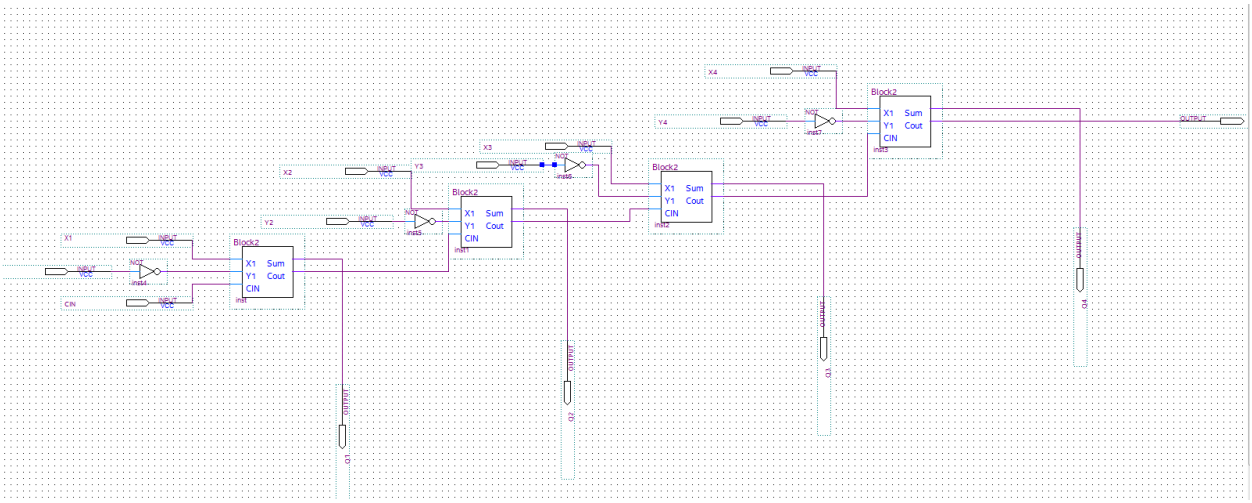

Figure 10: Complete Logic Diagram for 4-Bit Full Subtractor on Quartus Prime

4. 4 One-Bit Full Adder/Subtractor

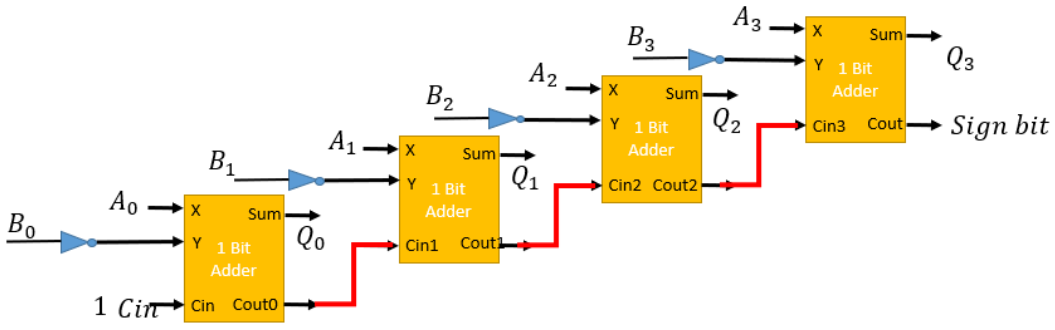

Figure 11: Block Diagram for the 4 One-Bit Full Adder/Subtractor

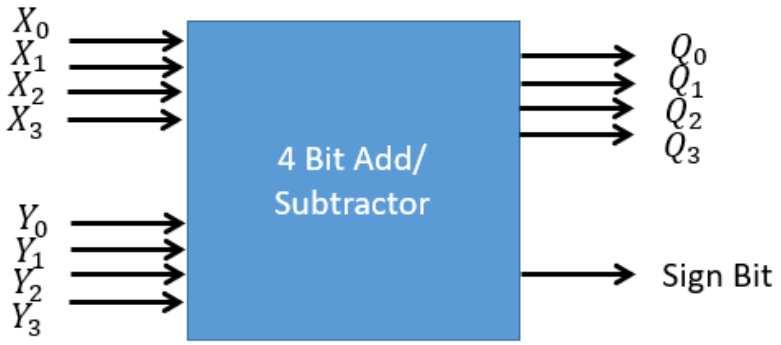

Figure 12: Block Diagram for the 4 One-Bit Full Adder/Subtractor (Simplified)

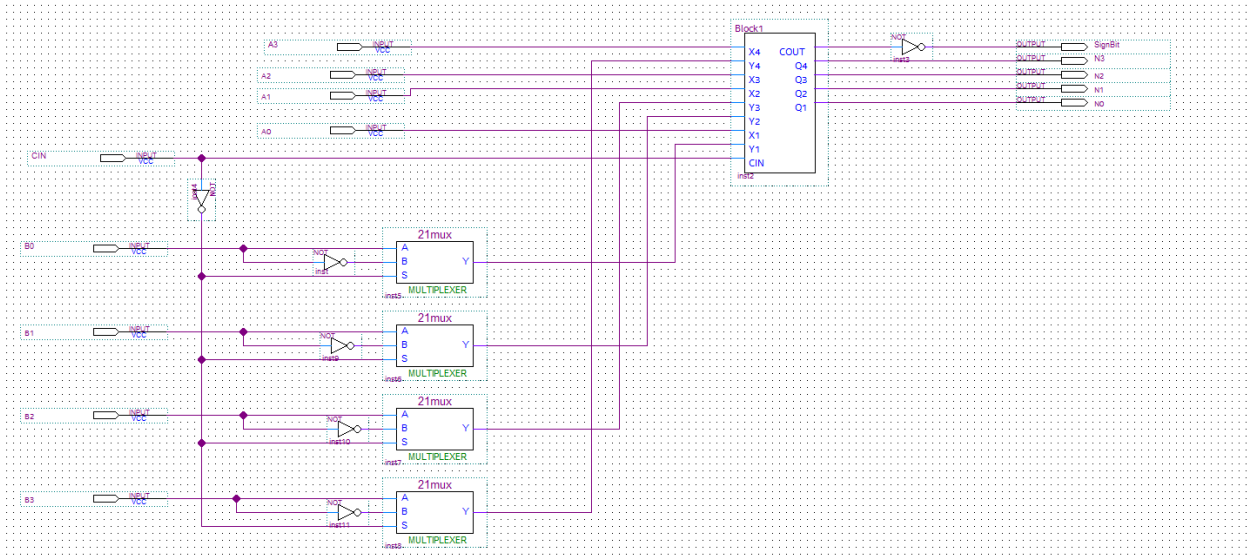

Figure 13: Complete Logic Diagram for the 4 One-Bit Full Adder/Subtractor

Results and Simulations

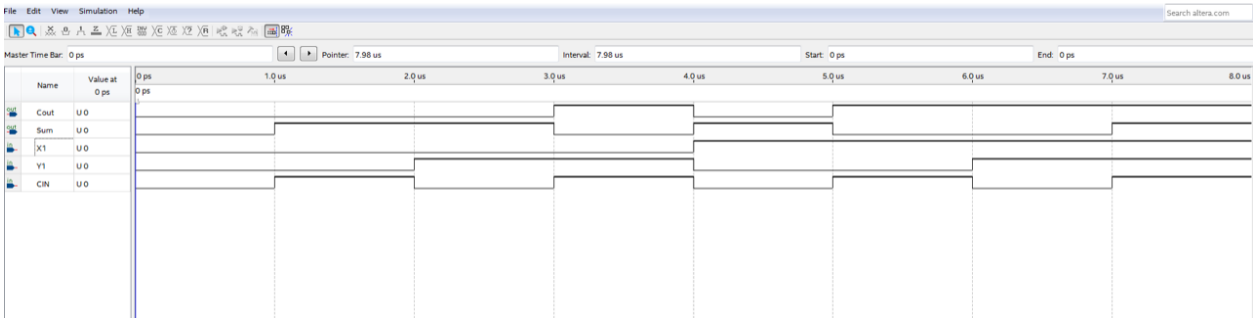

Figure 11:One-Bit Adder Simulation

Figure 11 shows the simulation results for the one-bit adder. It can be seen that the results directly reflect the values on the truth table in Table 2. The sum displays X + Y.

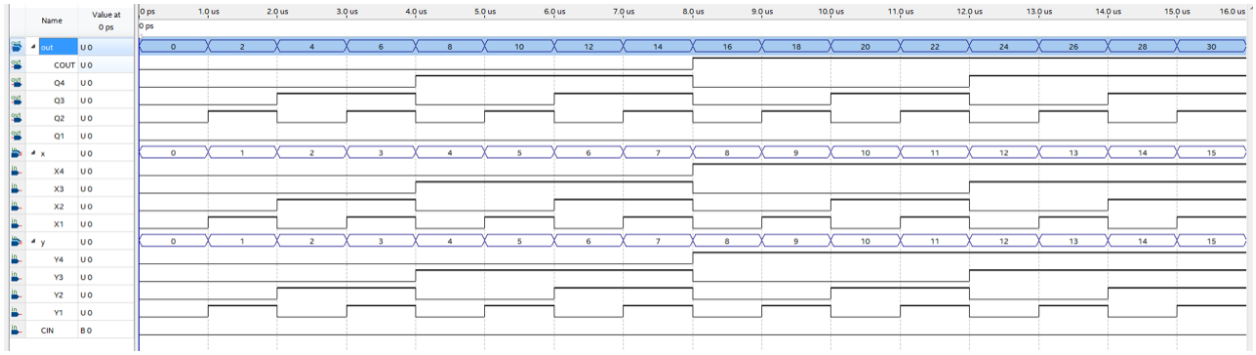

Figure 12: Four-Bit Adder Simulation

Figure 12 shows the simulation for the four-bit adder. It can be seen that when X is 4 in decimal and Y is 4 (decimal), the output is 8 (decimal) which is indeed 4+4. The four-bit adder takes two binary values and add them together to produce one binary number as it would have in decimal. The Cin value was set to 0 for the 4 Bit Adder.

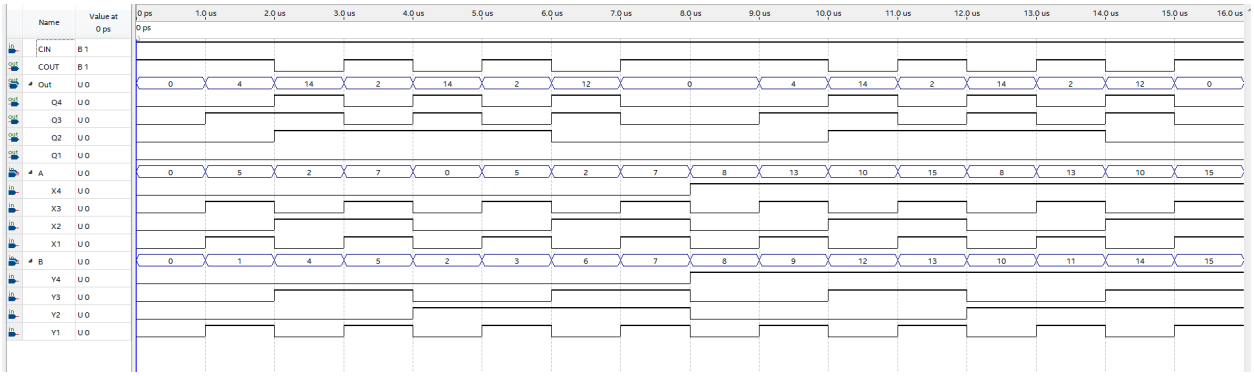

Figure 13: Four-Bit Subtractor Simulation

Figure 13 shows the subtraction of two four-bit numbers. It can be seen that when A equals 13 in decimal and B equals 9 in decimal, the difference between the numbers is 13-9 which equals 4 in decimal. The Cin value was set to 1 for subtraction.

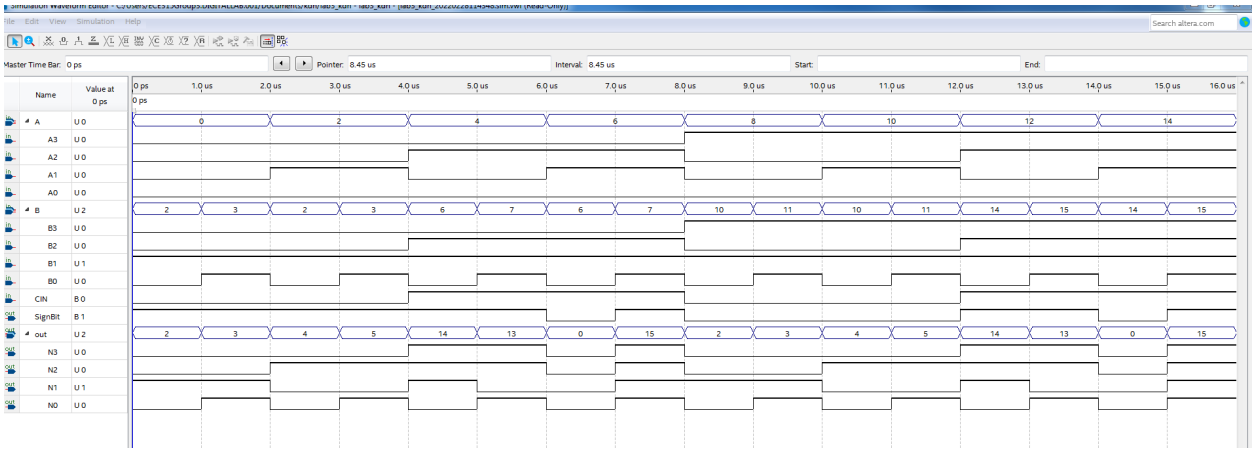

Figure 14: Four-Bit Adder/Subtractor Simulation

In Figure 14, it can be seen that when Cin is 1, subtraction will take place and whereas when it is 1, addition will take place. An example is: Cin=1, A=4 (decimal), B=6 (decimal) and A-B=-2 (decimal) which is 14 (decimal) in two's complement. However when Cin=0, A=2, B=2 (decimal) and A+B=4 (decimal). It should be noted that Cout is not equal to Sign bit. Sign bit is actually the complement of Cout.

Conclusion

The Four-Bit adder and subtractor was created by first designing a truth table and then karnaugh maps for the sum and carry-out values needed for a One-Bit adder. The boolean equations for Sum and Cout were derived from the karnaugh maps. They were implemented to create the logic diagram for the One-Bit adder. After creating one 1-Bit adder, three more were combined and connected in series to create a Four-Bit adder. In order to make a Four-Bit subtractor, four NOT gates were added to the second inputs in the Four-Bit adder and the value of Cin was forced high (or to 1). For the 4-Bit Adder/ Subtractor, four 2:1 Multiplexers were added to the 4-Bit Adder alongside six NOT gates in order to complete both addition and subtraction of 4-bit numbers (Max decimal number 15). This can be seen in Figure 13.

Since it was the second week of using Quartus Prime, we were still not completely knowledgeable about all the tools within the software and had trouble turning the one-bit adder into a symbol, but after researching we were able to successfully create it and save it into our library. This allowed for a much simpler implementation of the logic circuit. There were no limitations or flaws in the design, the design was able to successfully add and subtract 4-bit values. We learned how to use new tools and functions within the Quartus Prime software and these should prove to be very useful in the upcoming labs and future projects.