Lab 3 - Arithmetic Logic Unit

CEG2136

[redacted]

Table Of Contents

Introduction

The main objective of Lab 3 of Computer Architecture 1 is to help the students learn more of the functions of the Quartus II software while also implementing the knowledge gained from the material taught in class thus far. This Lab will focus on the LSC and an AC which help create an ALU. The meaning of these acronyms will be discussed further in this Lab Report. This Lab will also help develop our knowledge of the design hierarchy, which is crucial on Quartus II. Under the circumstances, Lab 3 will also help students become more familiarized with accessing VMware virtual machines and learning how to use it.

Requirements

The lab at hand has three major requirements; the ability to create a 4 bit logic and shift circuit (LSC), the ability to create 4 bit Arithmetic Circuit (AC), and the ability to combine these logic circuits and other circuits such as a full adder, to create a functioning ALU or Arithmetic Logic Unit. To integrate all of these separate circuits into one, the design hierarchy needs to be understood. The hierarchy of files to reach the goal of this project are expressed in Figure 1 below.

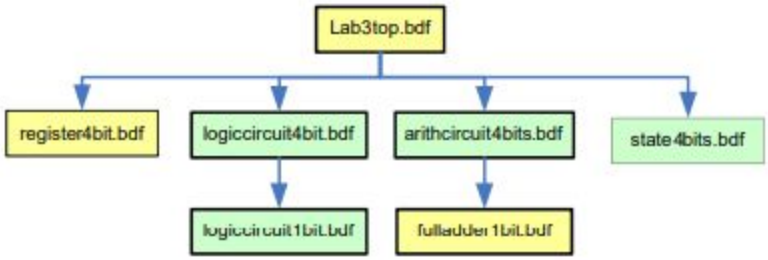


Figure 1. Hierarchy of Files

This hierarchy allows us to create an ALU datapath. The path our ALU should take is reflected below in Figure 2.

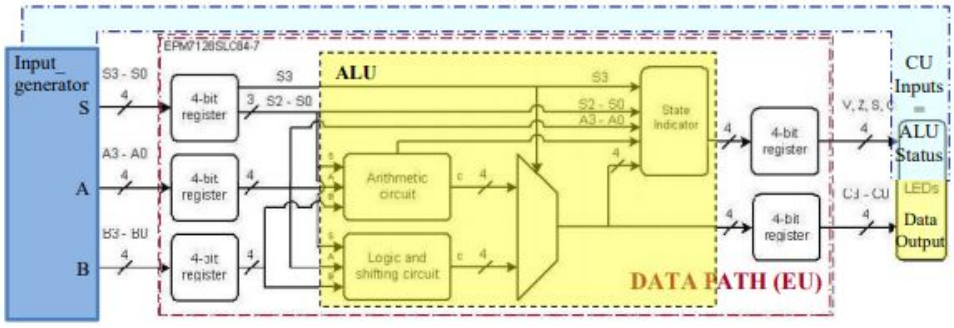
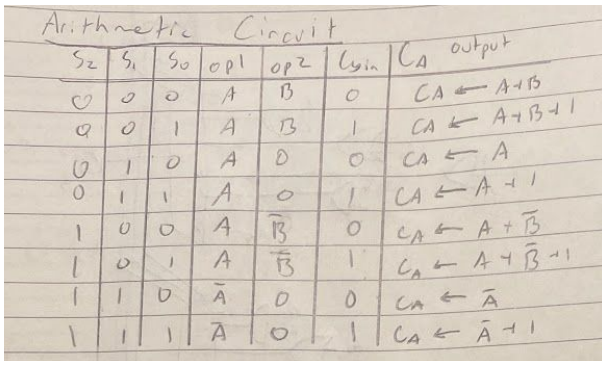


Figure 2. ALU Datapath

It is important to note that the LSC operates at the bit level meaning that it processes the data bit by bit while the AC is composed of full adders thus, it works in a more normal way by calculating outputs based on the operands.

Algorithmic Solution

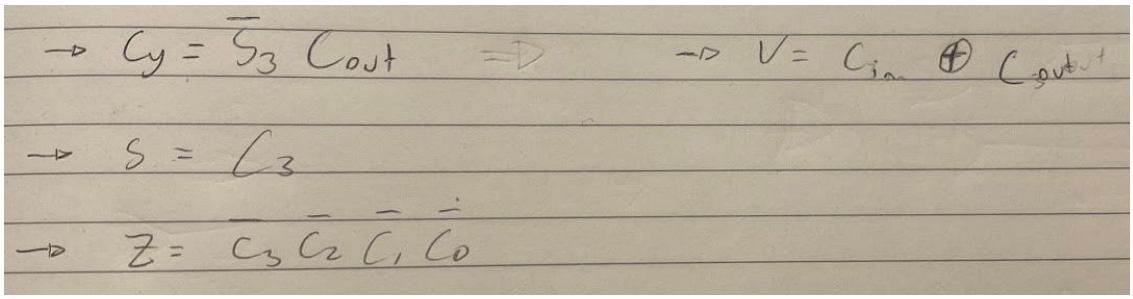
Though this Lab is not heavy on the alrogrimthics of Computer Architecture, we are still required to determine some values algorithmically, using a truth table, in order to create things like our AC, or the ALU stats registar which can be determined from the Lab Manual through logic equations. Figures 3 & 4 will reflect the equations that need to be deciphered and created.



Arithmetic Circuit

S_2	S_1	S_0	op1	op2	C_{in}	C_A output
0	0	0	A	B	0	$C_A \leftarrow A+B$
0	0	1	A	B	1	$C_A \leftarrow A+B+1$
0	1	0	A	0	0	$C_A \leftarrow A$
0	1	1	A	0	1	$C_A \leftarrow A+1$
1	0	0	A	$\bar{B}$	0	$C_A \leftarrow A+\bar{B}$
1	0	1	A	$\bar{B}$	1	$C_A \leftarrow A+\bar{B}+1$
1	1	0	$\bar{A}$	0	0	$C_A \leftarrow \bar{A}$
1	1	1	$\bar{A}$	0	1	$C_A \leftarrow \bar{A}+1$

Figure 3. AC Truth Table



$\rightarrow C_y = \bar{S_3} C_{out} \Rightarrow \rightarrow V = C_{in} \oplus C_{out}$

$\rightarrow S = C_3$

$\rightarrow Z = \bar{C_3}\bar{C_2}\bar{C_1}\bar{C_0}$

Figure 4. ALU Status Register Equations

Design

The following figures have been created using Quartus II or have been done during the prelab and are reflected in A.1, A.2, & A.3 of the Appendix. First the 4 bit register was created on Quartus using D Flip-Flops. After this diagram was made it was converted into a symbol so it could be used in other logic diagrams and integrated into our ALU

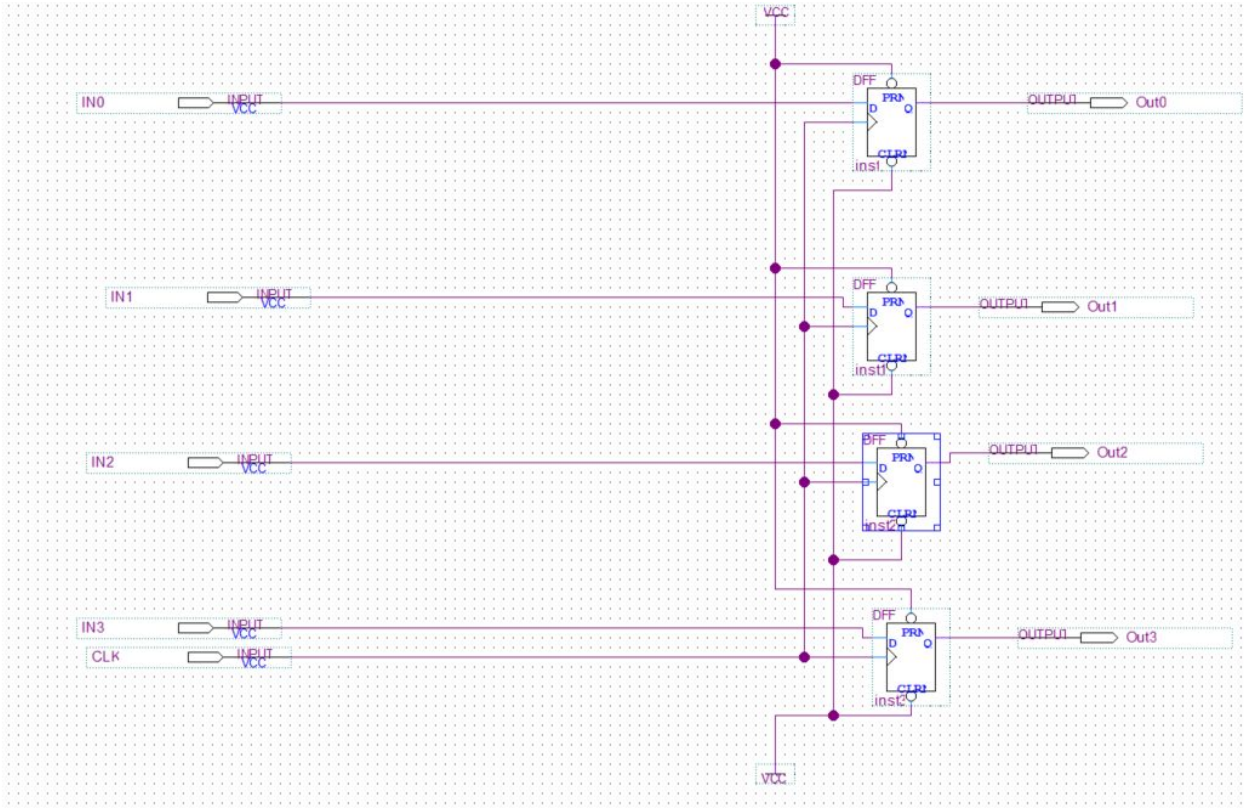


Figure 5. 4-bit Register Logic Diagram

Next, using our knowledge from the prelab, a 1 bit LSC was created and then converted to a symbol so that it could be used as a bit in helping to create a 4 bit LSC.

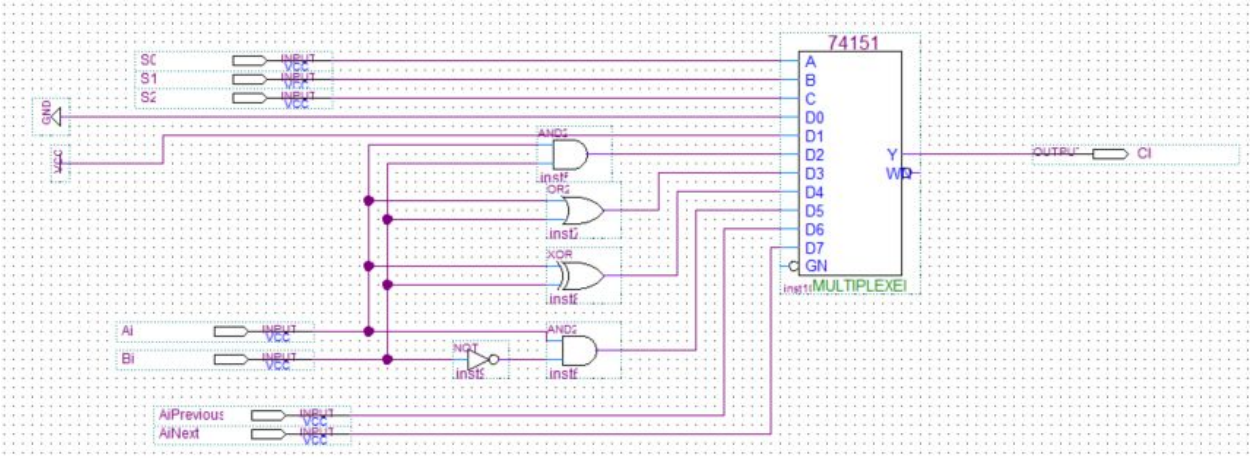


Figure 6. 1-bit Logic Diagram

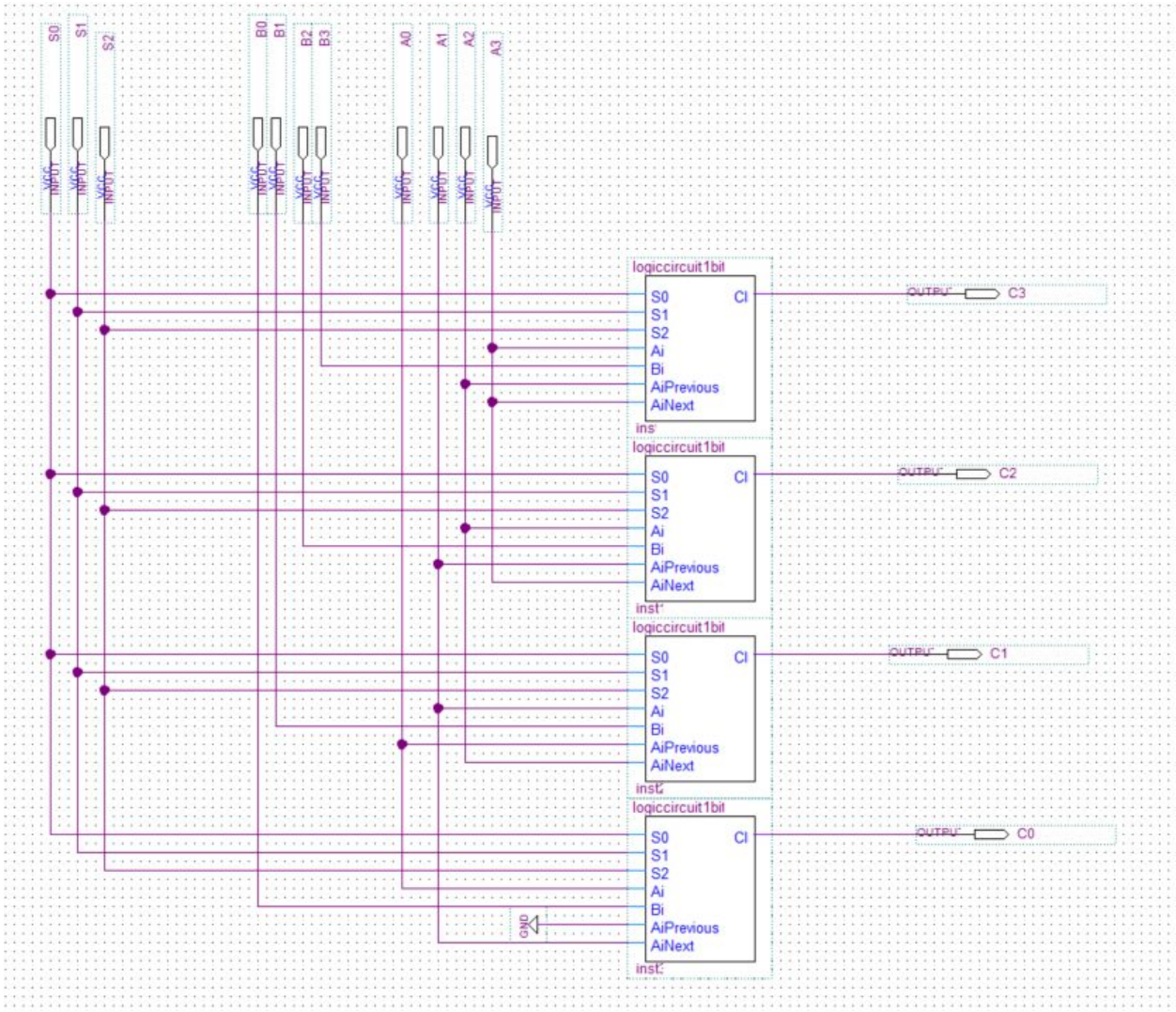


Figure 7. 4-bit Logic Diagram

With one crucial part of the ALU completed now, it was time to move onto creating the Arithmetic Circuit. As we saw in the prelab and in class, the AC is created using a full adder. The full adder is turned into a symbol and would be added as a component of the 1 bit AC. When looking at the prelab work done, this can be reflected in section A.2 of the Appendix.

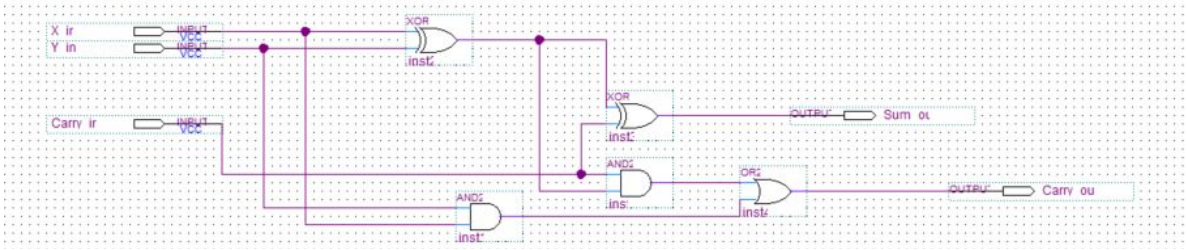


Figure 8. Full Adder Logic Diagram

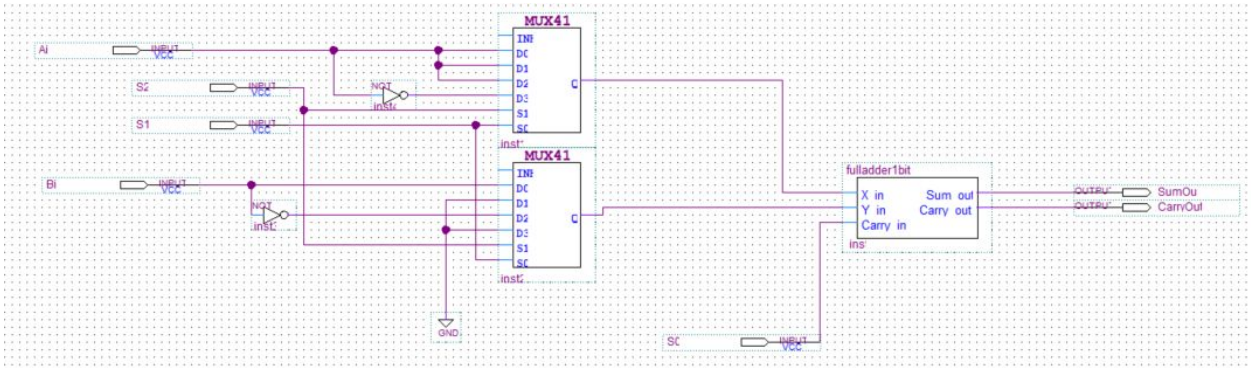


Figure 9. 1-bit AC Logic Diagram

Using the 1 bit AC, a 4 bit AC diagram can be constructed allowing us to have the second crucial part of the ALU created.

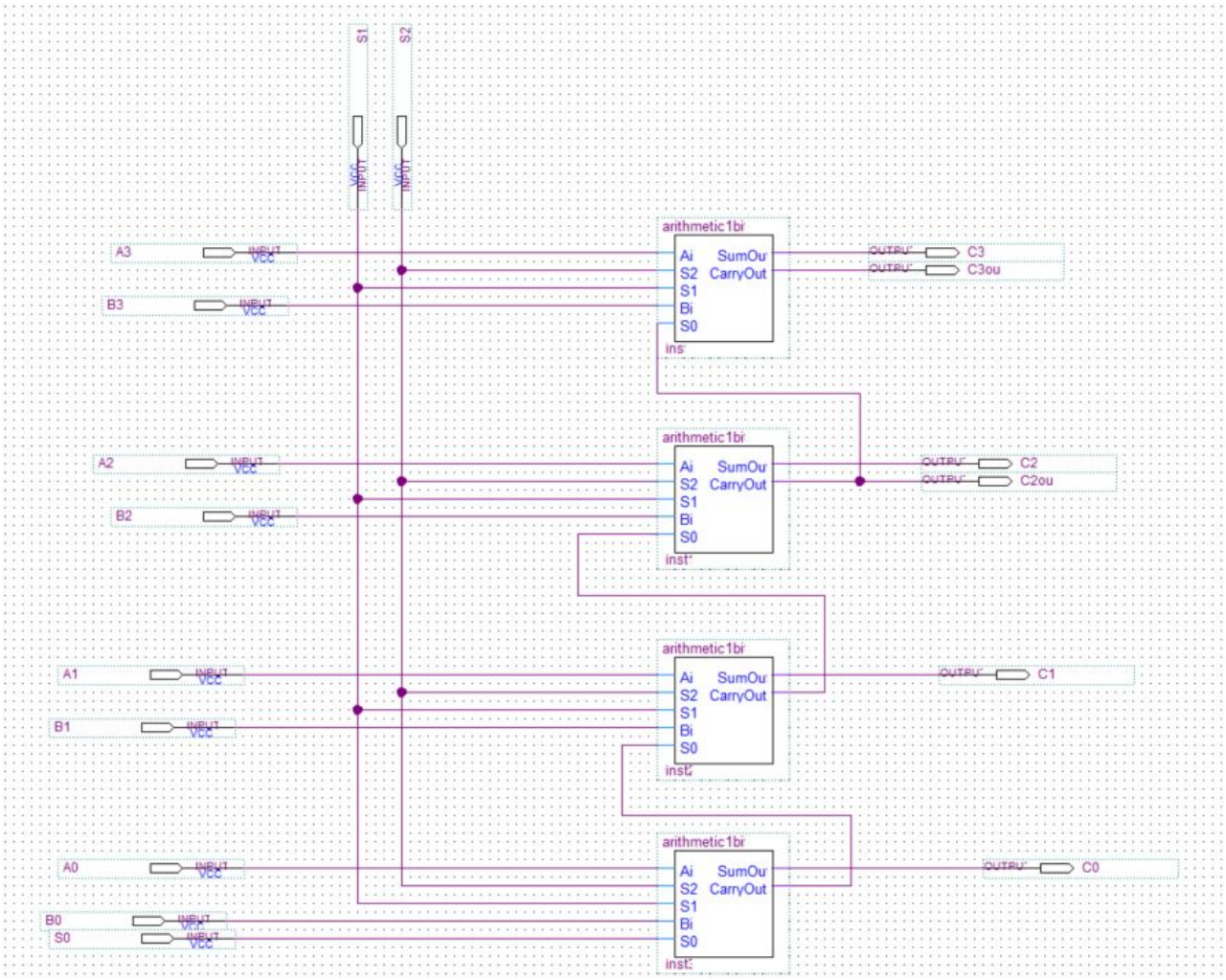


Figure 10. 4-bit AC Logic Diagram

Apart from the data outputs of the ALU as seen in Figure 2, another important final component needed to check on the ALU's correct function is the ALU status register. This register can be characterized by 4 bits; C, S, Z, & V. These bits represent the functions of Carry, Sign, Zero, and Overflow. In the pre lab a set of instructions for these bits are written out. These sentences are converted to logic equations, as shown through the prelab work done in A.3 of the Appendix, and then integrated into a logic diagram using the required amount of gates.

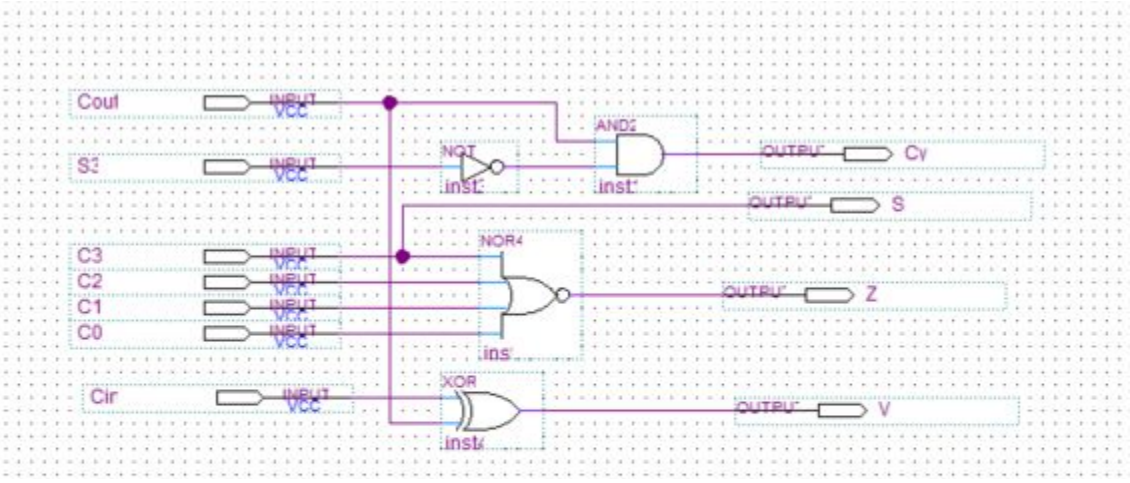


Figure 11. ALU Status Register Logic Diagram

Finally with all the components made and turned into symbols, a new block diagram labeled "Lab3Top" is set to the top of the hierarchy table and all the symbols are added in, along with a 74252 multiplexer. Thus, we have reached the data path of the ALU.

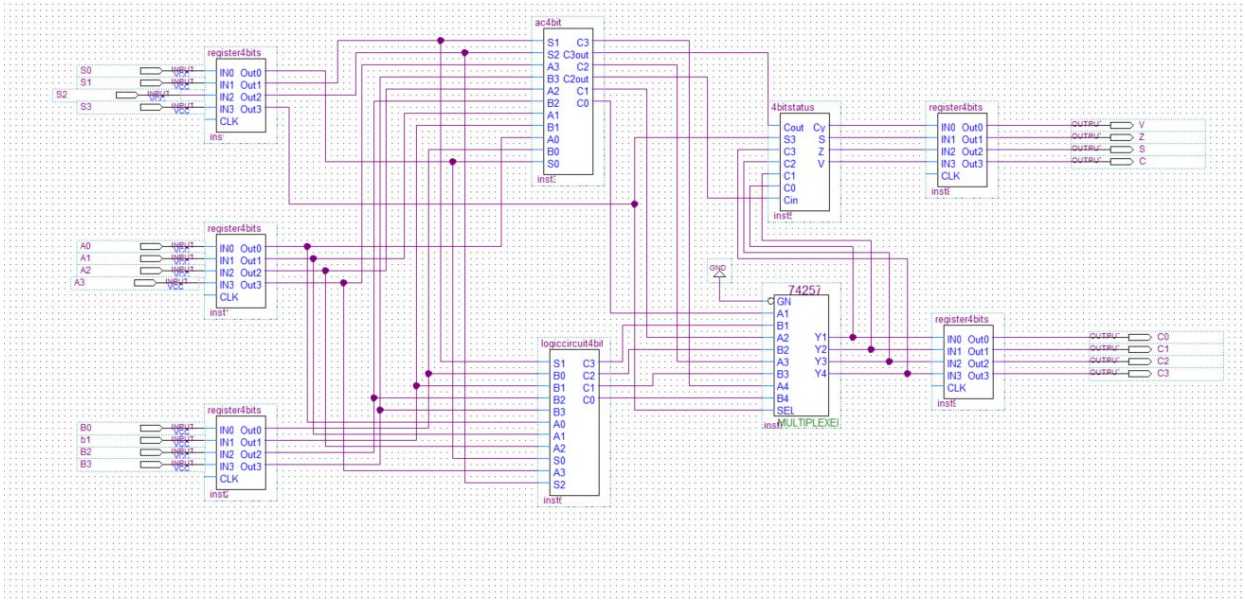


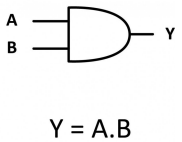
Figure 12. ALU Logic Diagram

Components Used & Implementation

For all the logic diagrams that were created during this lab, various components were required to be used. The following are all essential components needed.

AND

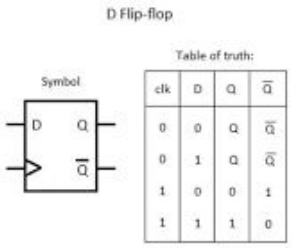
Inputs		Output
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1



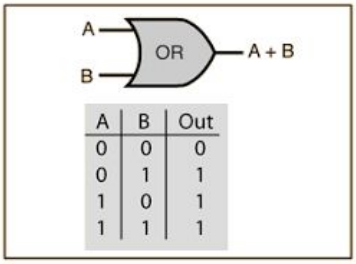
Y = A.B

AND Gate

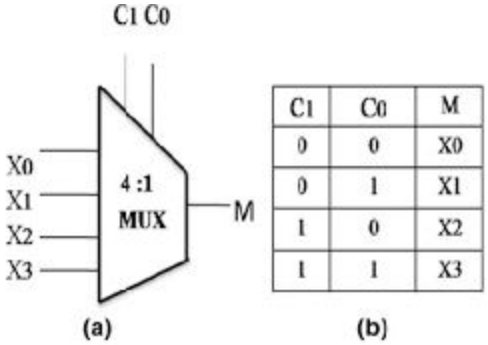
D Flip-Flops



OR



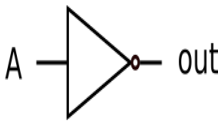
4x1 MUX



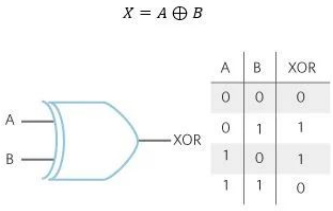
NOT

NOT:

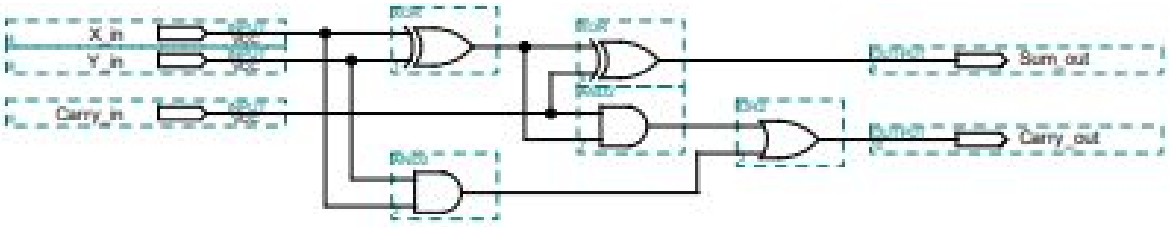
A	out
0	1
1	0



XOR



Full Adder



The implementation of all the logic diagrams was done at the end to create a ALU and this was done successfully as shown in Figure 12 below

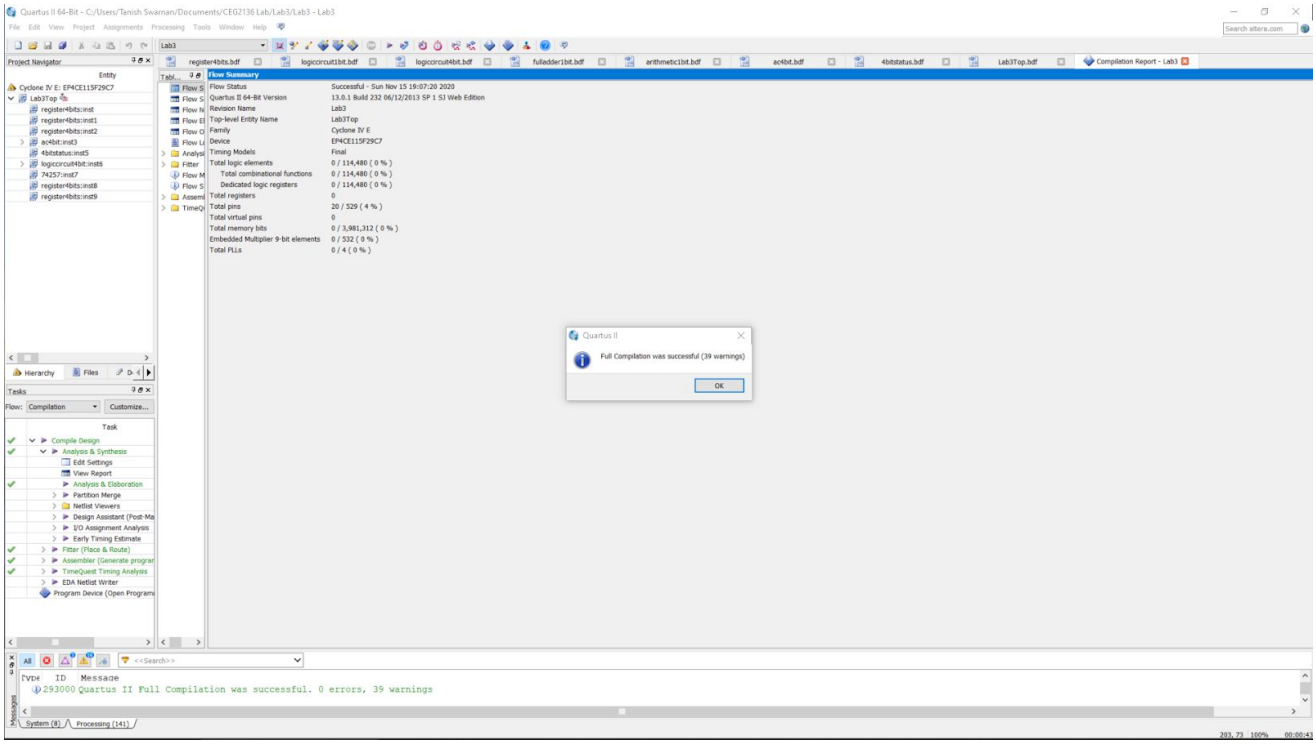


Figure 13. ALU Success

Simulation

The simulation was the part of the lab where I felt that my best work was not represented. Since the simulation is done virtually through a waveform diagram this year, extra steps had to be taken to create a functional simulation. Ultimately the simulation created, based on the logic diagram and instructions given in the Lab 3's manual, was not successful as I tried to debug my problem 4 or 5 times and I did not end up getting the correct values for A & B shown in the figures below.

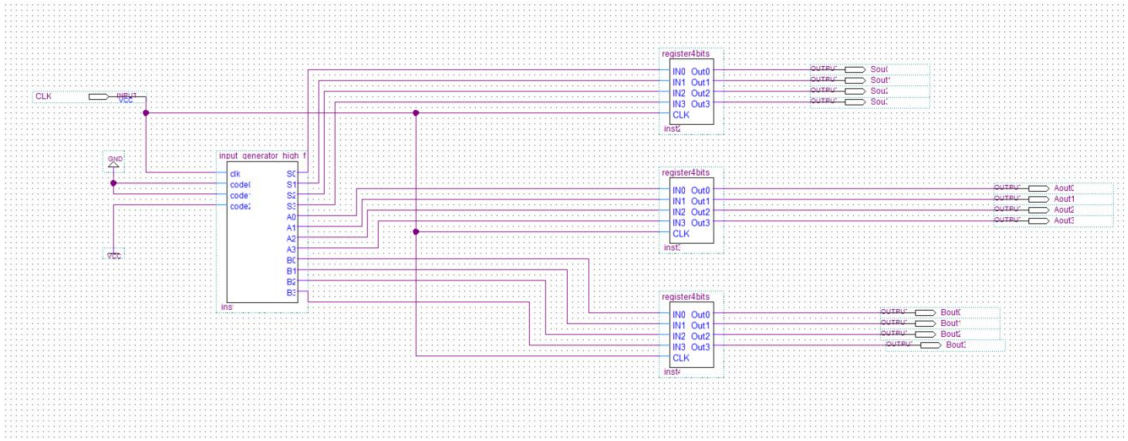


Figure 14. ALU Test Circuit

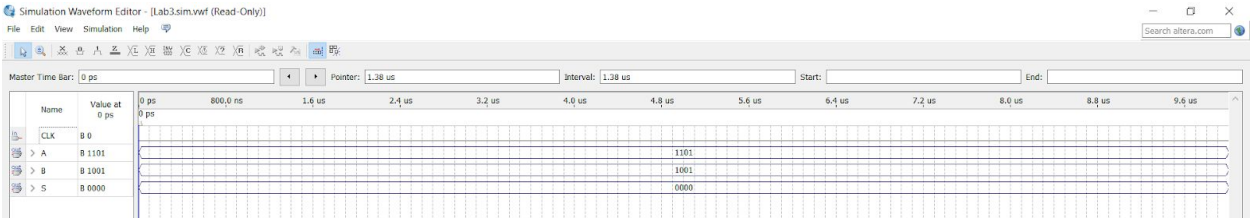


Figure 15. Waveform Results 1

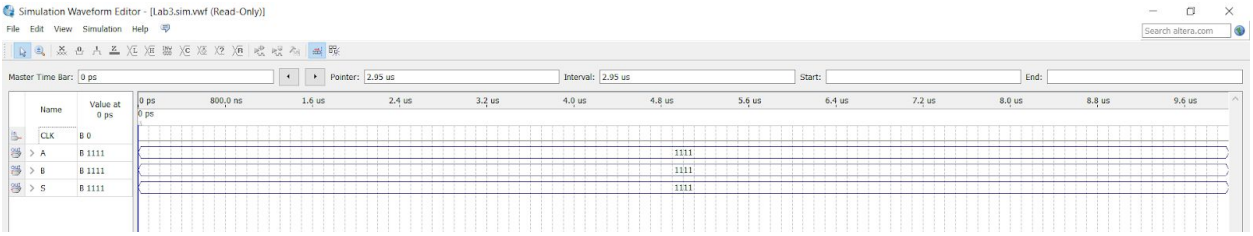


Figure 16. Waveform Results 2

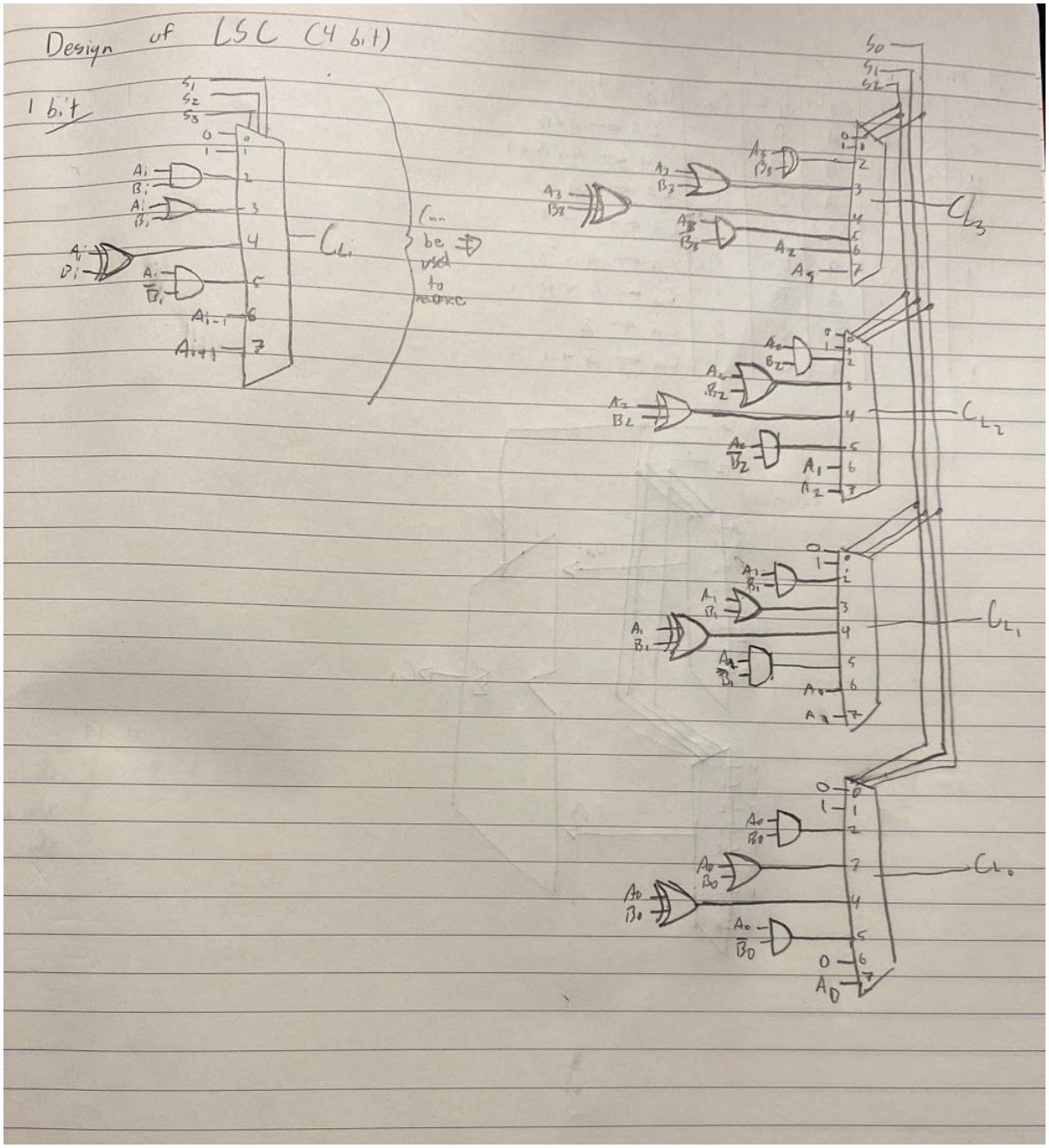
The set of first results were A=1101 & B=1001, which is not correct because the code I had was 110 and the results should have been A=1101 & B=1001. The second set of results do not make sense to me as I made some minor adjustments and got 1111 for everything.

Discussion & Conclusion

The problems that arose in this Lab with concerns to the simulations stem down to the fact that more practice is needed in this area of the course. Considering that the ALU is a critical part of this course and program, I will most likely be spending more time debugging and understanding the concepts to further improve my knowledge for the next Lab. Ultimately this lab gave me the opportunity to expand my knowledge of LSC, AC, and ALU and the Quartus II software. The experience gained in this lab will be beneficial for the next lab.

Appendix

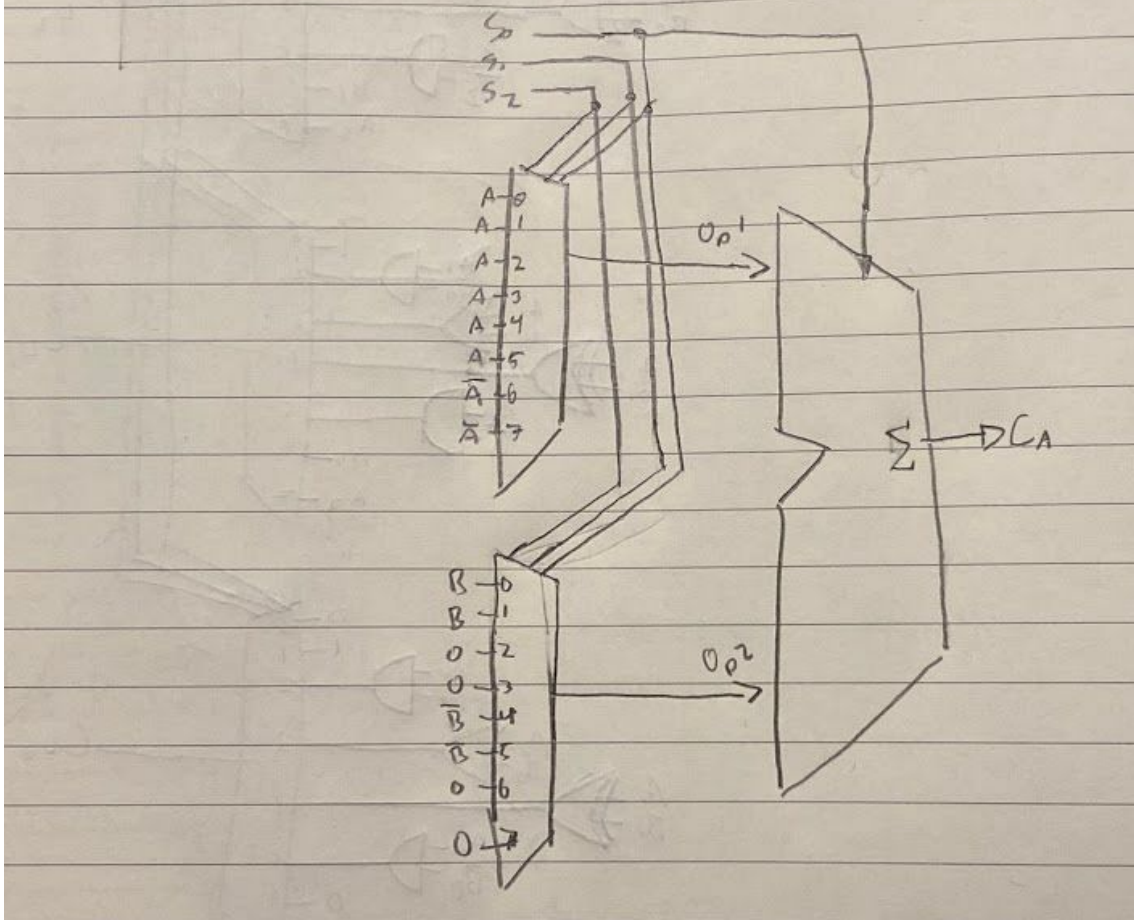
A.1 LSC



A.2 AC

Arithmetic Circuit

S_2	S_1	S_0	op1	op2	C_{in}	C_A output
0	0	0	A	B	0	$C_A \leftarrow A+B$
0	0	1	A	B	1	$C_A \leftarrow A+B+1$
0	1	0	A	0	0	$C_A \leftarrow A$
0	1	1	A	0	1	$C_A \leftarrow A+1$
1	0	0	A	$\bar{B}$	0	$C_A \leftarrow A+\bar{B}$
1	0	1	A	$\bar{B}$	1	$C_A \leftarrow A+\bar{B}+1$
1	1	0	$\bar{A}$	0	0	$C_A \leftarrow \bar{A}$
1	1	1	$\bar{A}$	0	1	$C_A \leftarrow \bar{A}+1$



A.3 ALU status register