Lab 3: Design and Simulation of Sequential Logic Circuits

Fall 2018

School of Electrical Engineering and Computer Science
University of Ottawa
Professor: Dr. Fadi Malek

Group 6
Khang Nguyen 300007277
Syed Haider Rizvi 6842303

Experiment Date: October 15 2018
Submission Date: November 12 2018

Introduction:

The goal of this lab is to create an arithmetic and logical unit (ALU). the ALU is found in a CPU and is in charge of making micro operations and connecting them into calculations and operations. The ALU we are creating will have 16 operations in total, 8 arithmetic equations and 8 logical operations.

Lab Problem:

During the lab we will be building an ALU. Since the ALU we are going to build contains 16 different operations, one of the important requirements to consider is how the 16 operations will be differentiated. In the table below, we can find the arithmetic and logical operations.

Table 1: Table of micro-operations to be completed by the ALU

Circuit	Control Word S3 S2 S1 S0	ALU Data Output	Micro-operation Description
	0 0 0 0	$C \leftarrow A + B$	Addition
	0 0 0 1	$C \leftarrow A + B + 1$	Add with carry
	0 0 1 0	$C \leftarrow A$	Transfer A
	0 0 1 1	$C \leftarrow A + 1$	Increment A
	0 1 0 0	$C \leftarrow A + \overline{B}$	Subtraction A - B with borrow (using 1's complement of B gives a result lower by 1 than a conventional subtraction).
	0 1 0 1	$C \leftarrow A + \overline{B} + 1$	Subtraction A - B (use 2's complement of B)
	0 1 1 0	$C \leftarrow \overline{A}$	NOT A (complement A)
	0 1 1 1	$C \leftarrow \overline{A} + 1$	2's complement of A
	1 0 0 0	$C \leftarrow$"0000"	Reset C
	1 0 0 1	$C \leftarrow$"1111"	Set C
	1 0 1 0	$C \leftarrow A \wedge B$	A AND B
	1 0 1 1	$C \leftarrow A \vee B$	A OR B
	1 1 0 0	$C \leftarrow A \oplus B$	A EXCLUSIVE-OR B
	1 1 0 1	$C \leftarrow A \wedge \overline{B}$	Reset A bits selected by "mask" B
	1 1 1 0	$C \leftarrow a\text{shl } A$	Shift A left (signed multiplication by 2)
	1 1 1 1	$C \leftarrow a\text{shr } A$	Shift A right (signed division by 2)

Objectives:

- Enter the design of each ALU components using Quartus II graphics editor
- Assign the input-output pins and prepare the design for downloading and testing on the Altera DE2-115 board

- Test the ALU:
    - Display the ALU outputs as binary values on LEDs
    - Using the simulator, record the waveforms at various states.

Solution:

To find the solution, we drew every component to then combine them into the final circuit. To start, we drew the 4-bit register (given in the lab PDF). the 4-bit register used D-type flip-flops.

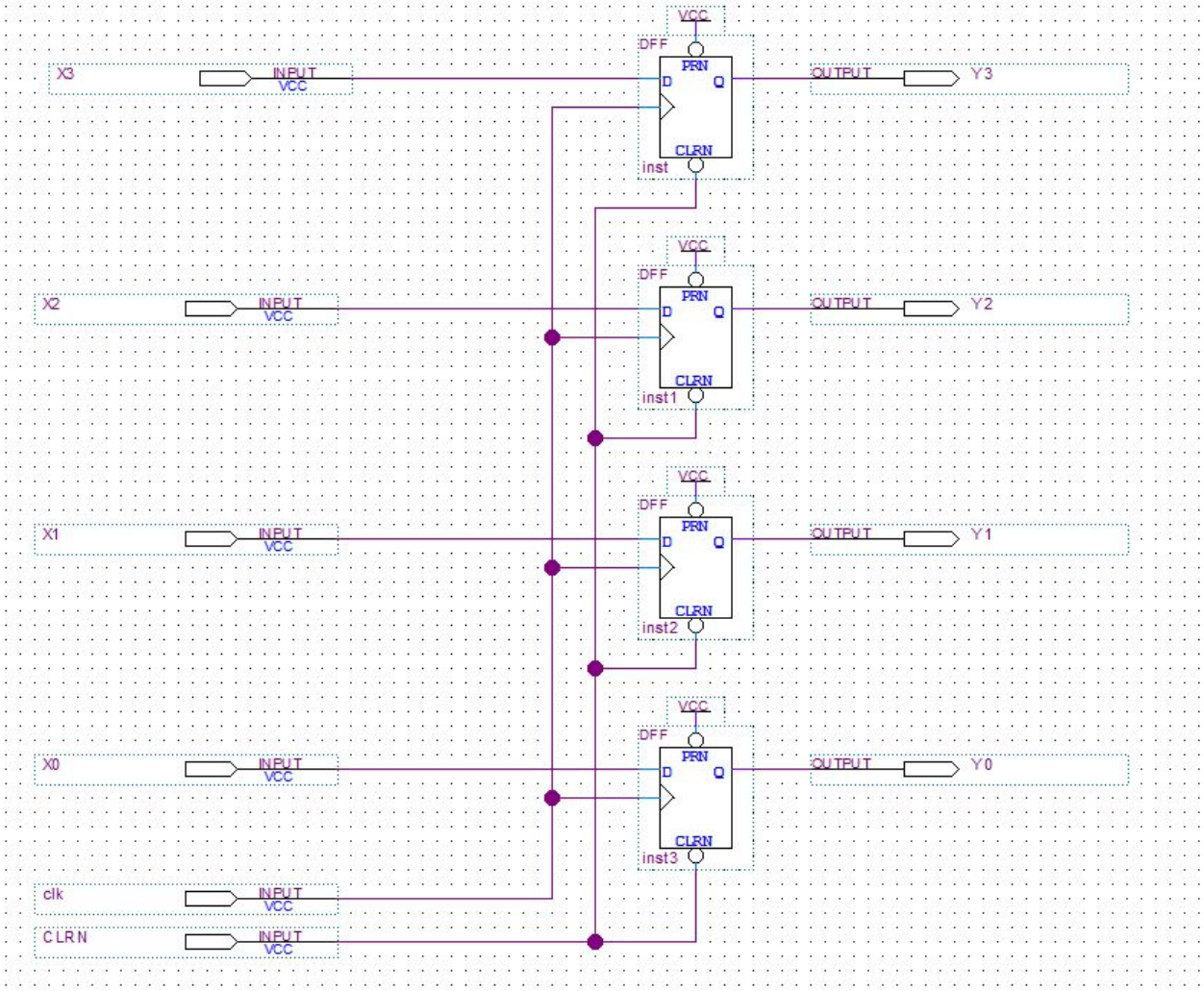


Figure 1: Screen capture of 4-bit register with D flip-flops

To continue we then decided to create the logic and shift circuit using a 8x1 multiplexer (multiplexer 74151). There will be 3 selector variables to choose between 0 and 7. Each one will be doing a different shift or logical operation.

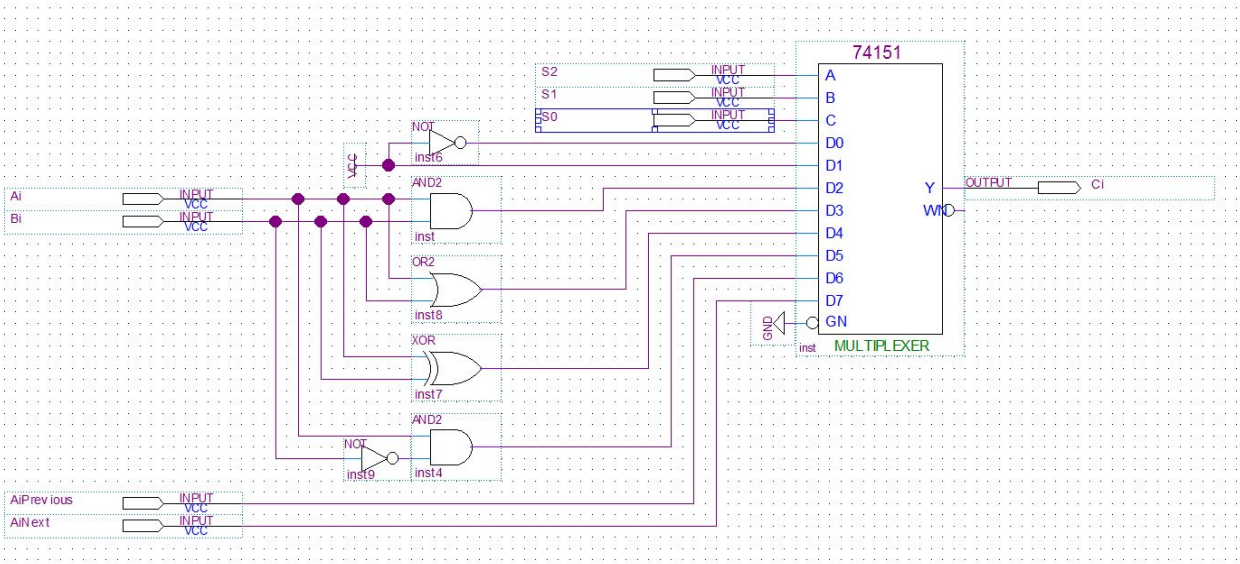


Figure 2: Screen capture of 1-bit logic and shift circuit with 8x1 multiplexer

To continue, we combined 4 logic and shift circuits to create a 4-bit logic and shift circuit. This is simply the same thing as a 1 bit logic and shift circuit but the inputs are across all 4 registers.

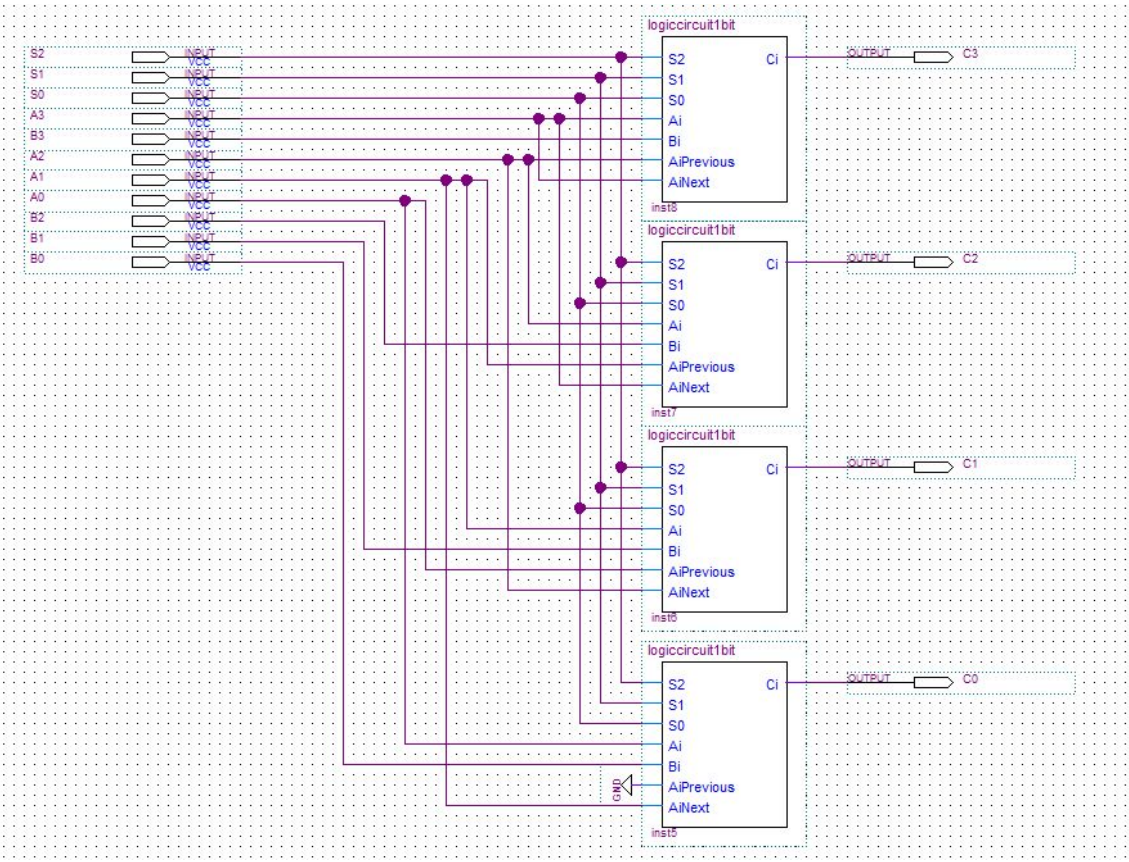


Figure 3: Screen capture of 4-bit logic and shift circuit with 4 logic and shift circuits

To continue, we then created the 1-bit full adder. The one bit adder simply adds x y and carryin and outputs sum out and carry out.

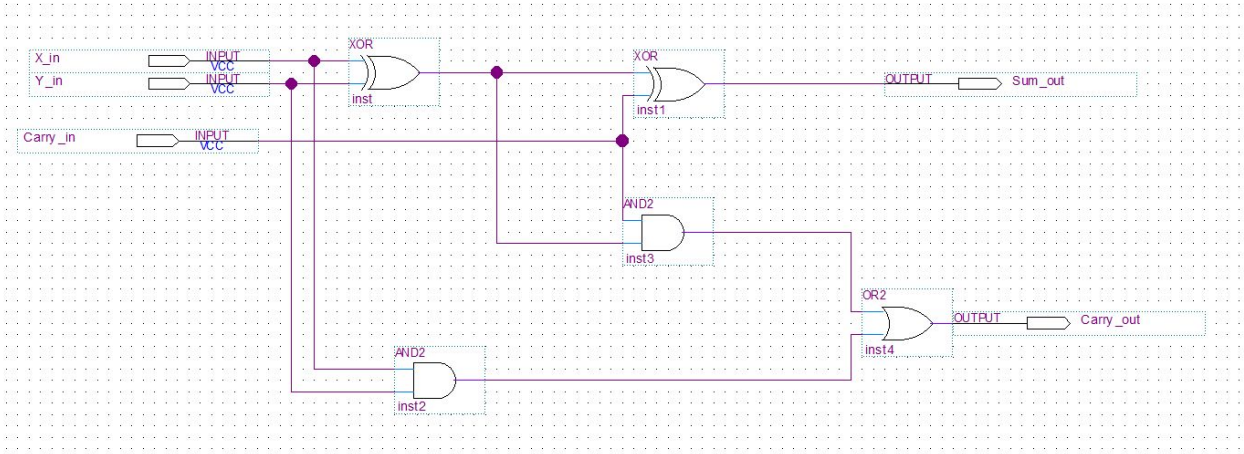


Figure 4: Screen capture of 1-bit full adder

We then created the 4-bit full adder which was the 1-bit full adder repeated 4 times. In which the selections were made through 3 selectors and 4 8x1 multiplexers. This ended up being more complex than the logic and shifting unit.

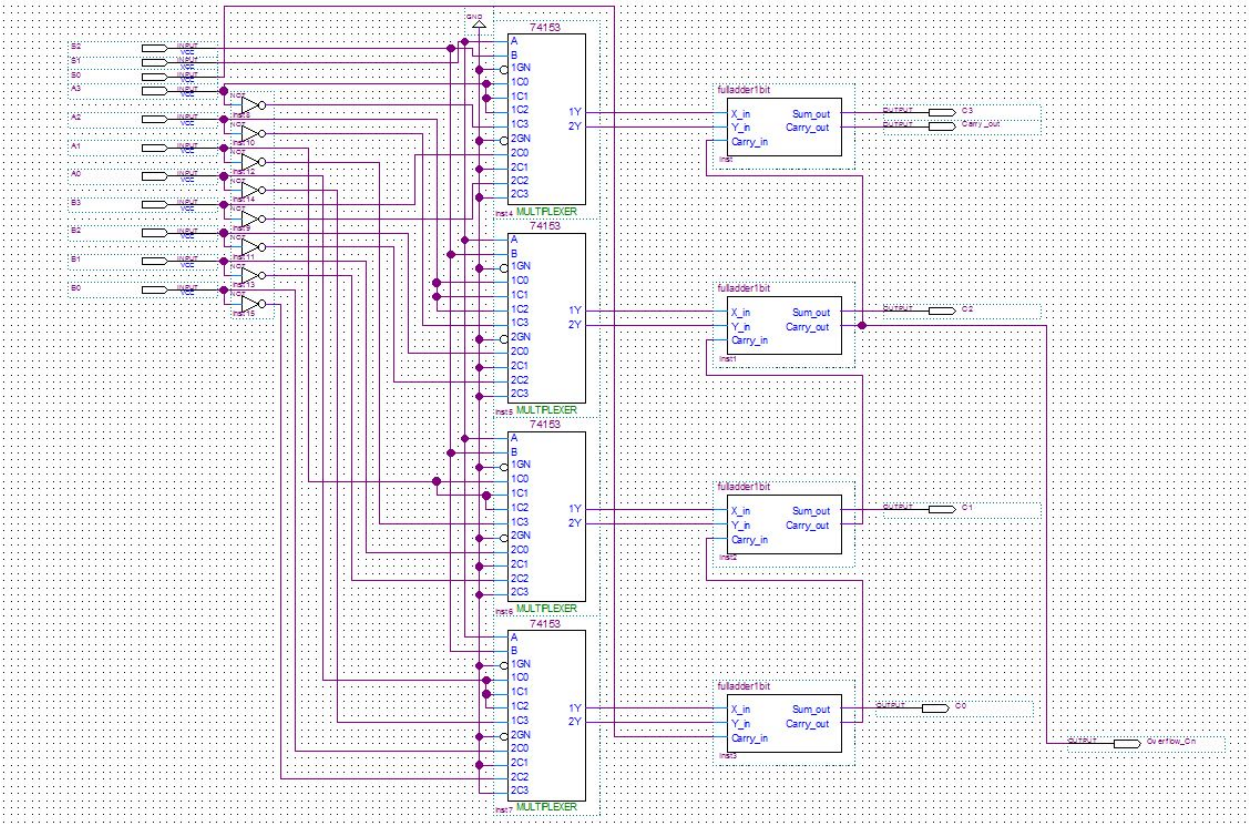


Figure 5: Screen capture of 4-bit full adder with 4 1-bit full adder and 4 8x1 multiplexers

Finally, we created the state indicator, the outputs are V, Z, S and Cy. V is 1 when there is an overflow during the arithmetic process. Z is 1 when C3, C2, C1 and C0 are all 0. S will be 1 when the C3 is also equal to one (most significant bit).

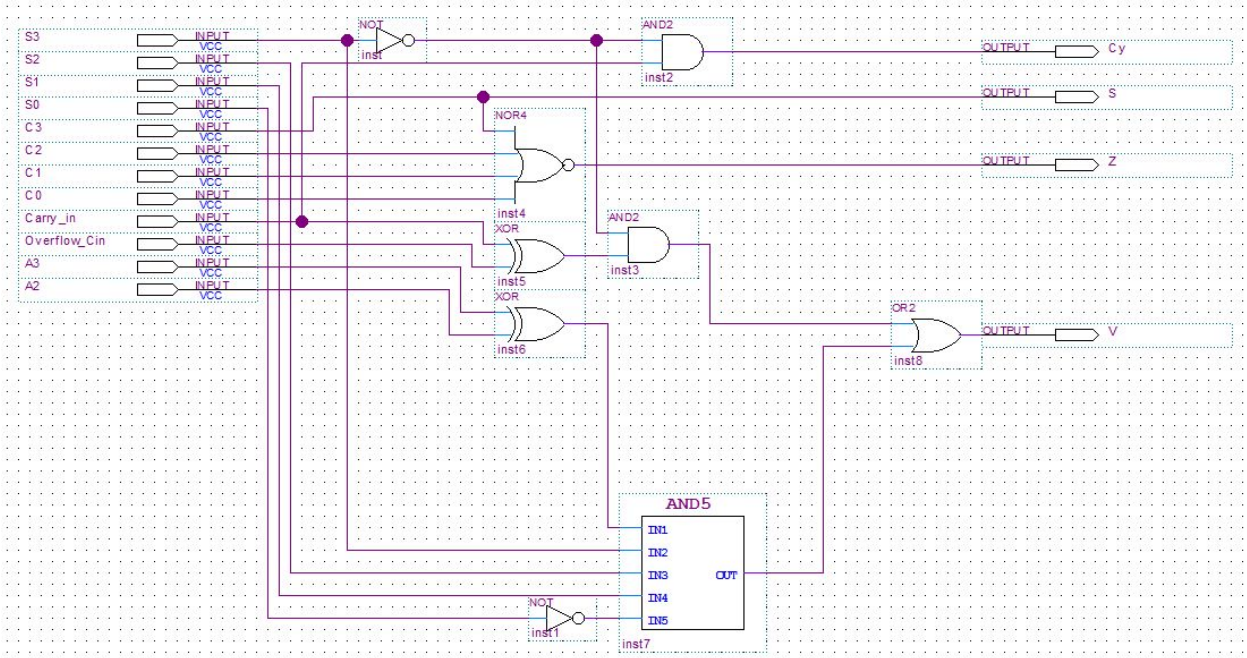


Figure 6: Screen capture of state indicator

Finally we implemented them all together to create an ALU.

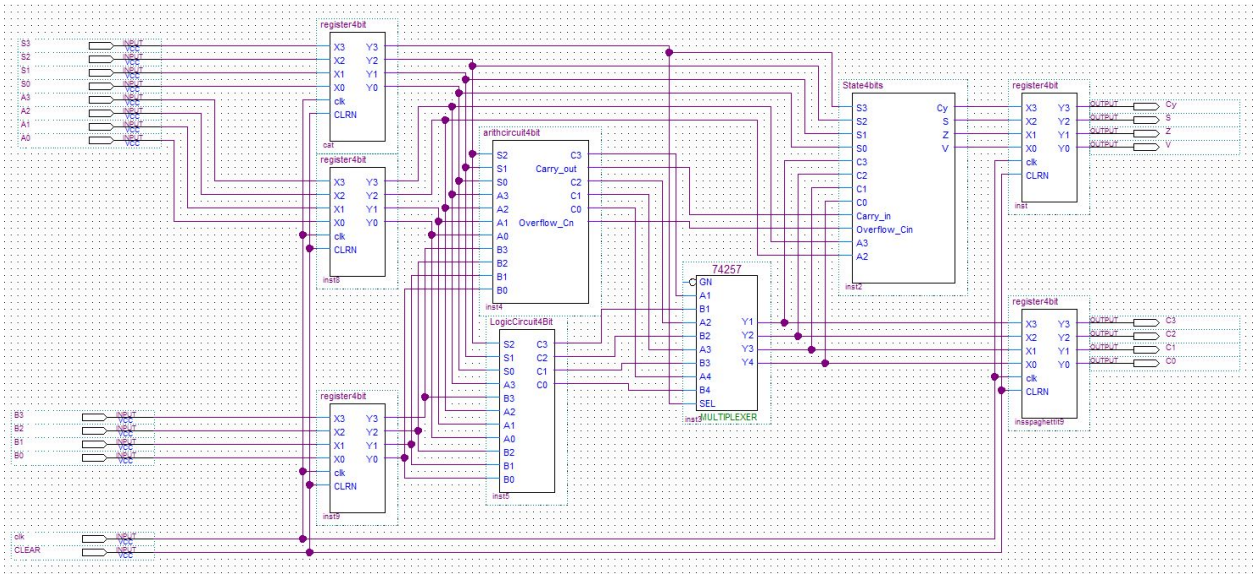


Figure 7: Screen capture of ALU

Discussion of used Components: The 4-bit register used D-type flip-flops. The 1-bit logic and shift circuit used one 8x1 multiplexer, while the 4-bit logic and shift circuit used 4 1-bit logic and shift circuit. The 1-bit full adder used XOR, OR and AND gates to effectuate the sum. The 4-bit full adder used 4 1-bit full adder and 4 8x1 multiplexers. Then the state indicator was made with AND5 gate, XOR, NOT, AND and OR gates. Finally the circuit had 4 selector inputs S3, S2, S1 and S0, it had 4 A and B inputs: A3, A2, A1 and A0, then B3, B2, B1 and B0. W then had 4 state outputs: Cy, S, Z, V. And finally 4 outputs: C3, C2, C1, C0.

Discussion of Actual Solution: With all the components in the ALU working together, the outputs were the ones from *Table 2*. They were approved by the Teaching Assistant and we were able to conclude that our designs were well made and had correct outputs.

Discussion of Tool: The tools we used were the Quartus II software and card. During the simulation process we were able to determine if we had derived a correct table. In the simulation (figure 8). Every 1 and 0 are indicated by the bumps up and down. With this method we were able to see that our pre-lab was correct. With the simulation, we were also able to display our results.

Discussion of Challenging Problems: A very challenging problem we faced during the laboratory was the combination of elements to create the ALU. We could not figure out why our XOR and left shit were not working, however we were able to determine that the cause was that in our 1-bit logic and shift circuit we had inverted the S2,S1 and S1. for that reason when testing each component directly we saw no problem, but when we connected it all together the selections going into the 4-bit logic and shift circuit where inverted.

Simulation and Verification:

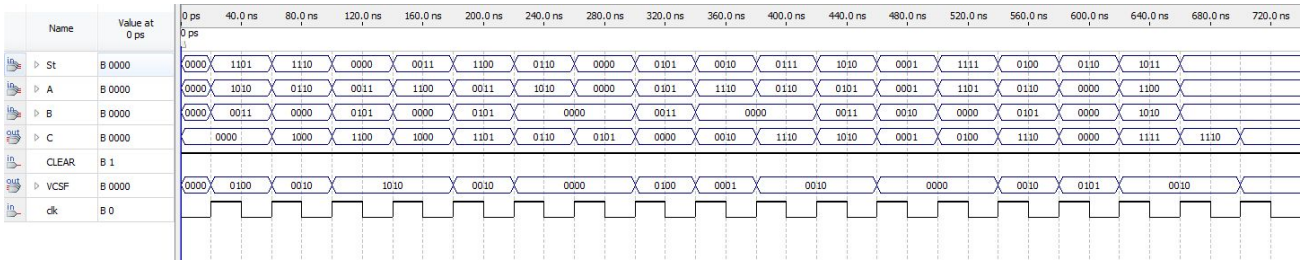


Figure 8: Simulation of ALU micro-operations

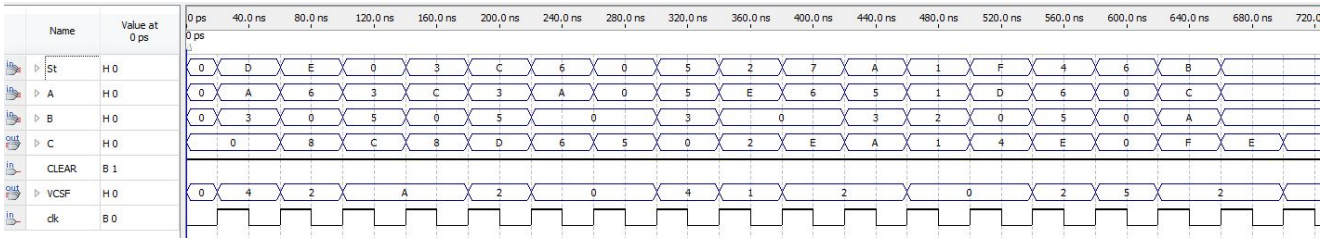


Figure 8: Simulation of ALU micro-operations (hexadecimal values)

Table 2: Experimental Data Observed from the simulation

CLK Cycle	Micro-operations	s3	s2	s1	s0	Aop	Bop	c	VZNCy	S16	A16	B16	C16	St16
1	C ← A ^ B’	1	1	0	1	1010	1100	1000	0100	D	A	3	8	D
2	C ← ashl A	1	1	1	0	0110	xxxx	1100	0010	E	6	0	C	E
3	C ← A + B	0	0	0	0	0011	0101	1000	1010	0	3	5	8	0
4	C ← A + 1	0	0	1	1	1100	xxxx	1101	1010	3	C	0	D	3
5	C ← A ⊕ B	1	1	0	0	0011	0101	0110	0010	C	3	5	6	C
6	C ← A’	0	1	1	0	1010	xxxx	0101	0000	6	A	0	5	6
7	C ← 0000	0	0	0	0	xxxx	xxxx	0000	0000	0	0	0	0	0
8	C ← A + B’ +1	0	1	0	1	0101	1100	0010	0100	5	5	3	2	5
9	C ← A	0	0	1	0	1110	xxxx	1110	0001	2	E	0	E	2
10	C ← A’ + 1	0	1	1	1	0110	xxxx	1010	0010	7	6	0	A	7
11	C ← A ^ B	1	0	1	0	0101	0011	0001	0010	A	5	3	1	A
12	C ← A + B + 1	0	0	0	1	0001	0010	0100	0000	1	1	2	4	1
13	C ← ashr A	1	1	1	1	1101	xxxx	1110	0010	F	D	0	E	F
14	C ← A + B’	0	1	0	0	0110	1010	0000	0101	4	6	5	0	4
15	C ← 1111	0	1	1	0	xxxx	xxxx	1111	0010	6	0	0	F	6
16	C ← A v B	1	0	1	1	1100	1010	1110	0010	B	C	A	E	B

To continue we were also able to load our circuit onto the ALTERA board to get it checked by the TA, we were able to check each actual output. We were able to assign the pins using the next table.

Table 3: Pin assignment for the final circuit

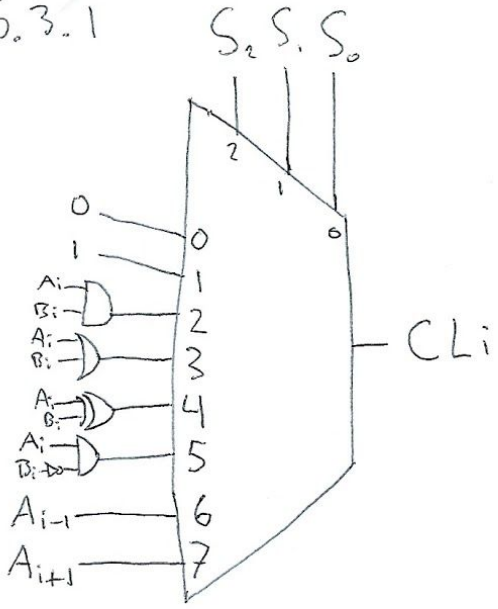
Signal	Pin Location	Component
S3	PIN_Y23	SW17
S2	PIN_Y24	SW16
S1	PIN_AA22	SW15
S0	PIN_AA23	SW14
A3	PIN_AD27	SW3
A2	PIN_AC27	SW2
A1	PIN_AC28	SW1
A0	PIN_AB28	SW0
B3	PIN_AB26	SW7
B2	PIN_AD26	SW6
B1	PIN_AC26	SW5
B0	PIN_AB27	SW4
C3	PIN_F21	LEDR3
C2	PIN_E19	LEDR2
C1	PIN_F19	LEDR1
C0	PIN_G19	LEDR0
V	PIN_E24	LEDG3
Z	PIN_E25	LEDG2
S	PIN_E22	LEDG1
Cy	PIN_E21	LEDG0
CLK	PIN_Y2	50 MHz oscillator
CLRN	PIN_AA24	SW13

Experimental verification of Operation of the Circuit: As seen above, our theoretical data was a match to the experimental data, achieved by loading our circuit onto the Altera board and testing each combination, meaning that our design was correct and the outcome expected. Our simulation followed a path that was described in our pre-lab.

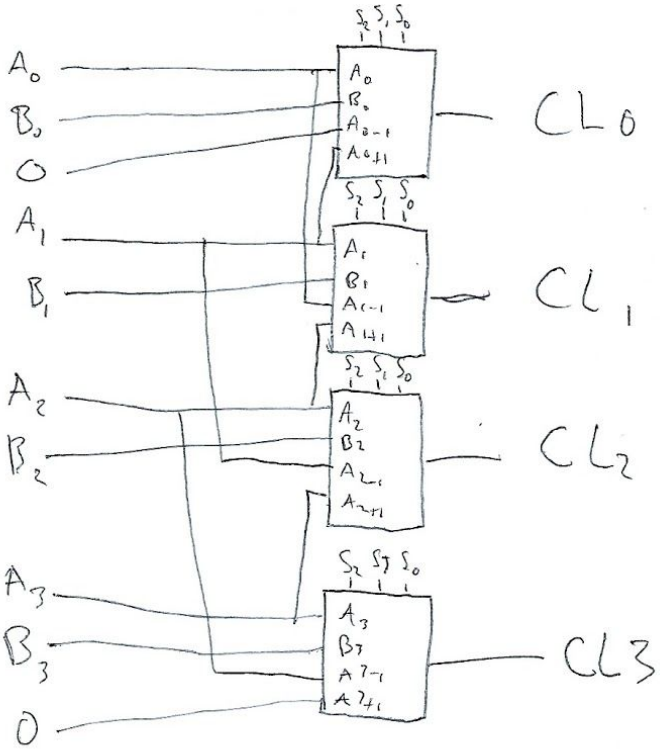
Conclusion: This lab did not contain many major problems for us, as the ALU functioned as we had expected it to as shown in Table 2. Initially, some of the logic operations in the ALU did not work as we had theorized, but these were due to minor errors in the connection of our 1-bit logic circuit, and were easily corrected. This lab taught us how to construct ALU circuits, and demonstrated how they can implement both arithmetic and logic functions.

Appendix:

5.3.1



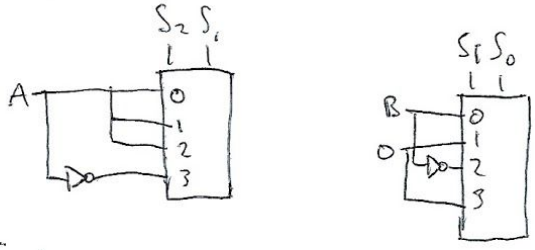
5.3.2



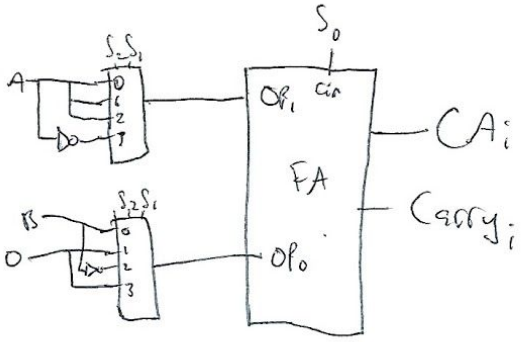
5.4.1

S_2	S_1	S_0	Op^1	Op^2	CyIn	CA output
0	0	0	A	B	0	$CA \leftarrow A+B$
0	0	1	A	B	1	$CA \leftarrow A+B+1$
0	1	0	A	0	0	$CA \leftarrow A$
0	1	1	A	0	1	$CA \leftarrow A+1$
1	0	0	A	$\overline{B}$	0	$CA \leftarrow A+\overline{B}$
1	0	1	A	$\overline{B}$	1	$CA \leftarrow A+\overline{B}+1$
1	1	0	$\overline{A}$	0	0	$CA \leftarrow \overline{A}$
1	1	1	$\overline{A}$	0	1	$CA \leftarrow A+1$

5.4.2



5.4.4



5.5

$Cy = Carry\ Out \cdot S_3'$

$S = C3$

$Z = [C0 + C1 + C2 + C3]'$

$V = Carry\ out + Carry B' + carry\ out' \cdot C02 = carry\ out \oplus carry\ 3$