Santa Clara University

Lab #6: Mini-calculator with Small Arithmetic Logic Unit (ALU)

ELEN 21L 51306 Tuesday 2:15-5pm

May 16, 2017

Group 2

Erika Sweet

Yutong Li

Lab #6: Mini-calculator with Small Arithmetic Logic Unit (ALU)

I. Objectives:

- Design and test a small adder/subtractor/logic unit (ALU)
- Use your ALU in a mini-calculator with one memory location.

II. Introduction:

In this lab, we used hierarchical design to incorporated the use of multiple lower-level symbol files into our final ALU design. The mini-calculator will use a small Adder/Subtractor Logic Unit described in the given statement to perform arithmetic operations on 4-bit values set using switches. We will also need a push button to save the current arithmetic result. After adding the push button, the upgraded mini-calculator allows the saved value to be used as one of the operands.

III. Procedure:

Part 1 - Design and test the small ALU

1. We built a 2to1 mux (as shown in Figure 1) and a 8to1 mux (as shown in Figure 2) based on the 2to1 mux and create a symbol for each of them to use later.

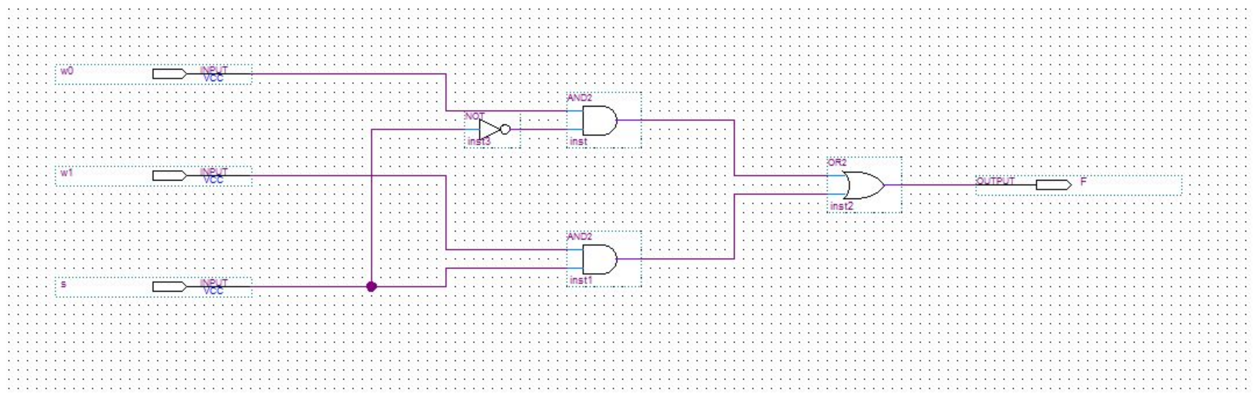

Figure 1. 2to1MUX

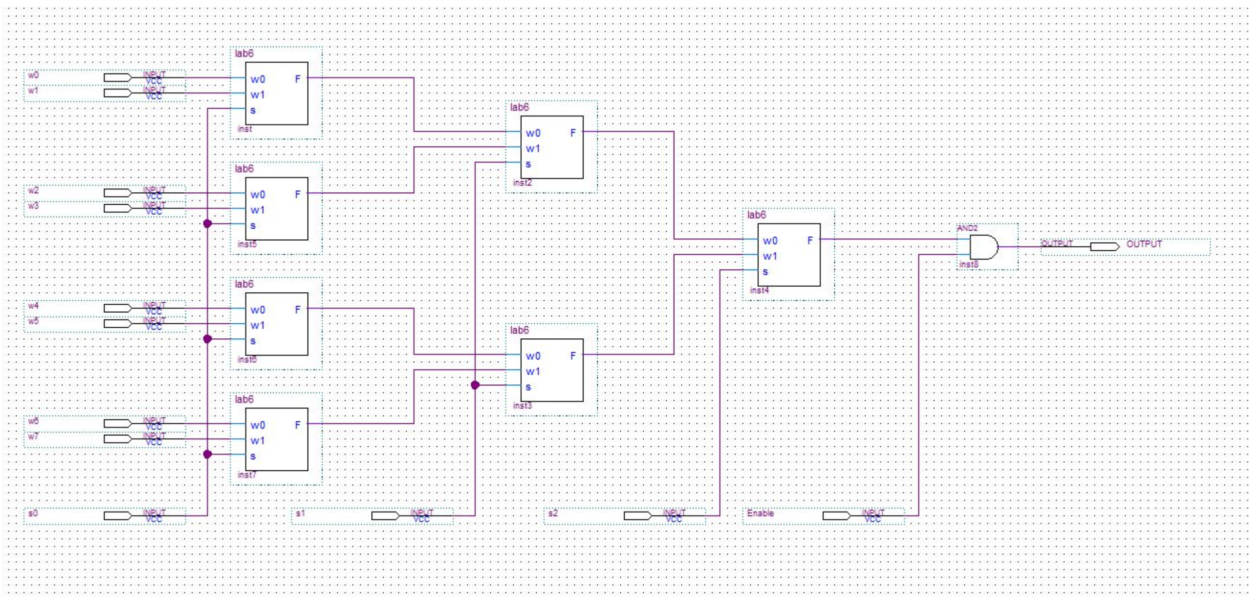
Figure 2. 8to1MUX

2. We built a symbol named MUX4 (as shown in Figure 3, 4, and 5) in which we included 4 8to1 mux. MUX4 will function as a mux in which the output will be a 4-bit input based on the switches.

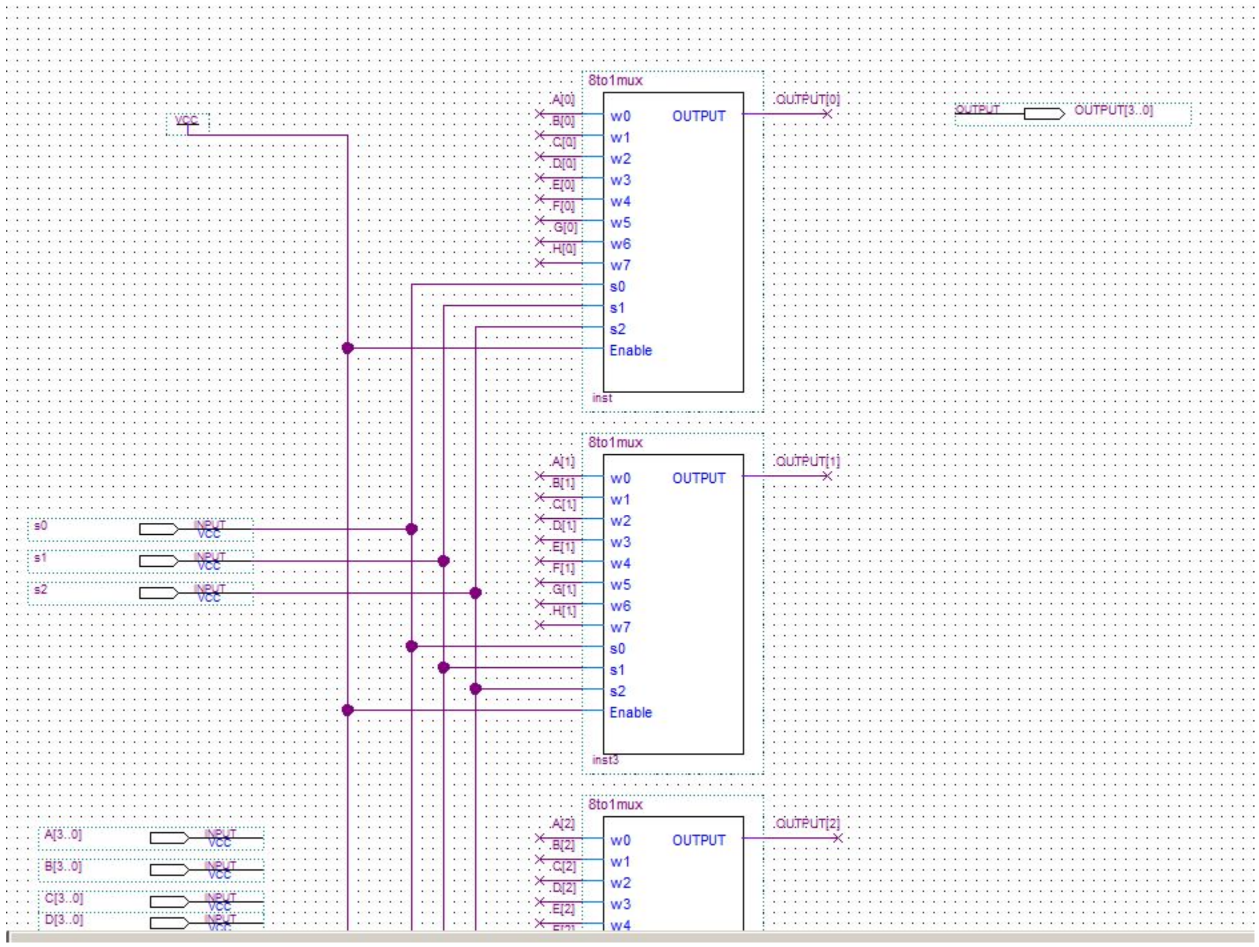

Figure 3. Detail of MUX4 (upper part)

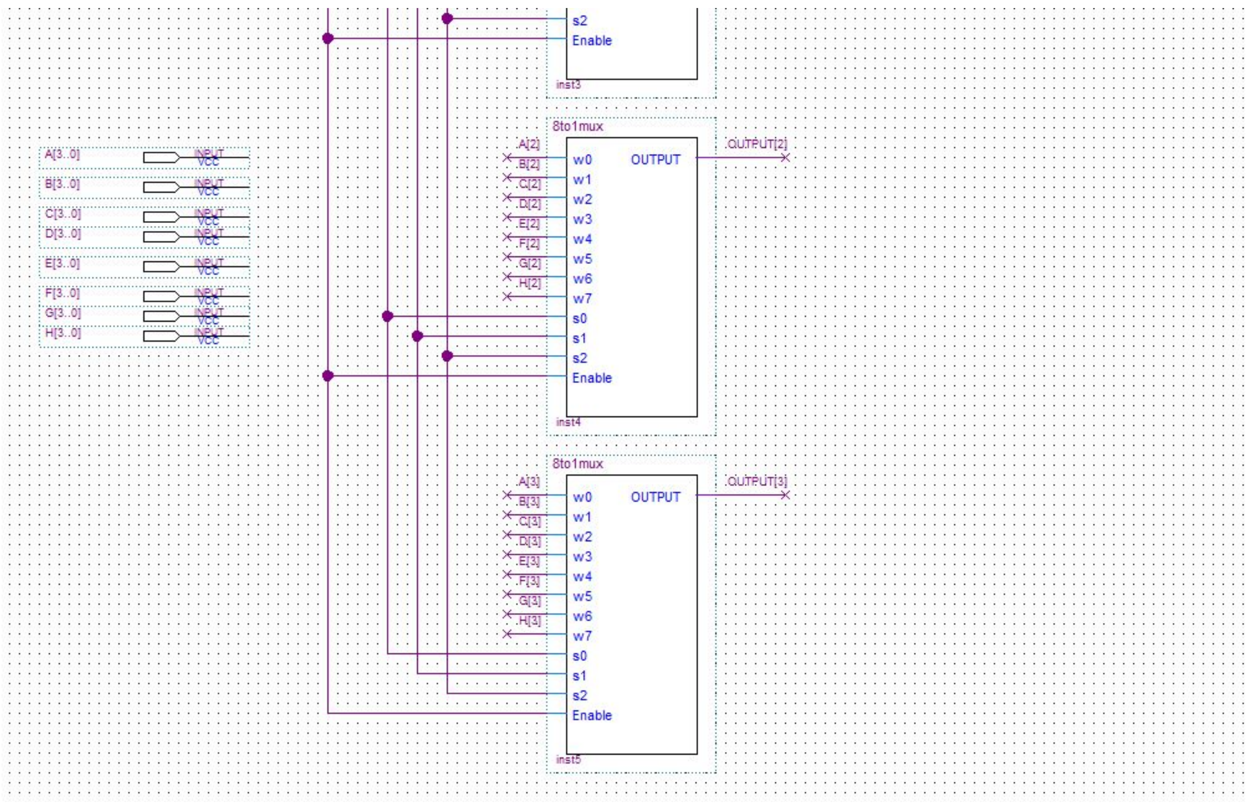

Figure 4. Detail of MUX4 (lower part)

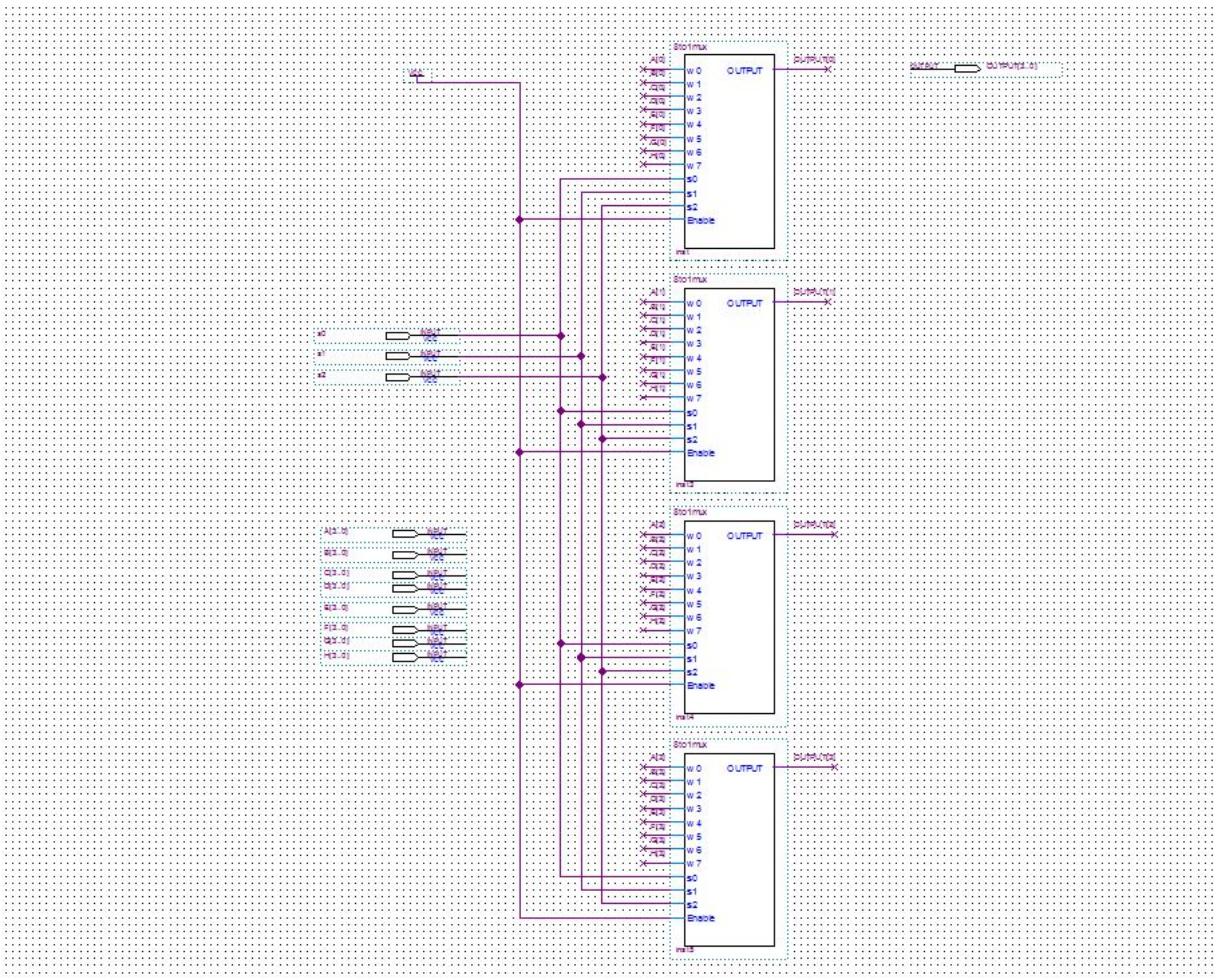

Figure 5. MUX4

3. We created a symbol name RCA (as shown in Figure 6) based on the circuit that we created in last lab.

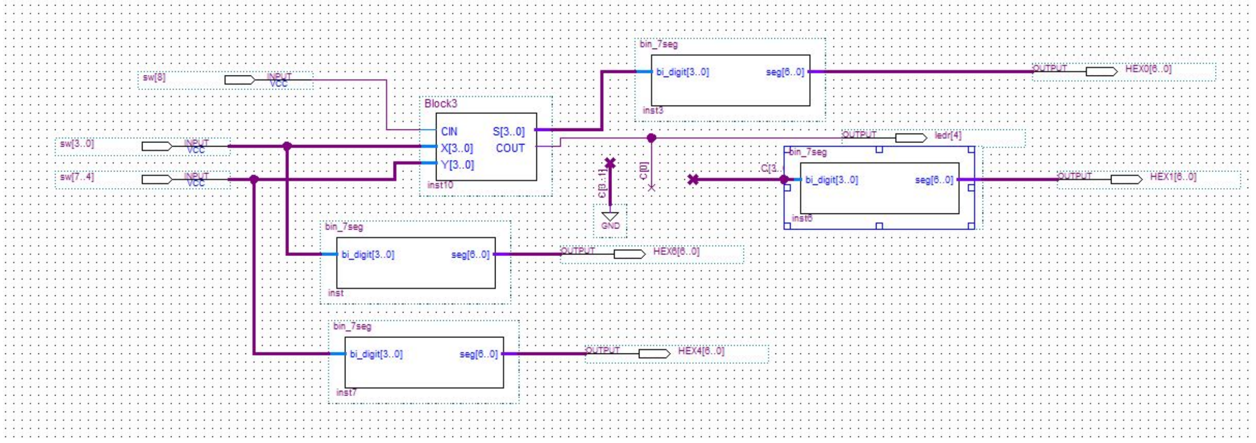

Figure 6. RCA

4. Based on the table (as shown in Figure 7) that we finished in prelab, which decides what the inputs and carry-in should be for each of the 8 operation, we creatde a circuit that will operate

according to the table, and called it ALU (as shown in Figure 8). ALU will takes in two 4-bit inputs, one is A[3..0] and the other is the OUTPUT[3..0] which is the output from MUX4 which is based on the combination of the three switches. We also build a symbol called inverter (as shown in Figure 9) which will invert every bit of the input. The carry-in which decides which combination of the 4-bit input B that we want. These two 4-bit inputs and the carry-in will goes into the RCA and the outputs will be the 7-segment displays of A, B and the result of the operations, and an LED light to indicate the state of COUT.

Op-Select input	Adder 4-bit A inputs	Adder 4-bit B inputs	Adder carry-in	Operation	Notes
r1 r0 p1 p0	**a3 a2 a1 a0**	**b3 b2 b1 b0**			
0 0 0 0	a3 a2 a1 a0	b3 b2 b1 b0	0	X = A + B	Add
0 0 0 1	a3 a2 a1 a0	b3 b2 b1 b0	1	X = A + B + 1	Add and increment
0 0 1 0	a3 a2 a1 a0	b3’ b2’ b1’ b0’	0	X = A – B – 1	Subtract and decrement
0 0 1 1	a3 a2 a1 a0	b3’ b2’ b1’ b0’	1	X = A - B	Subtract

Figure 7. Completed table from pre-lab.

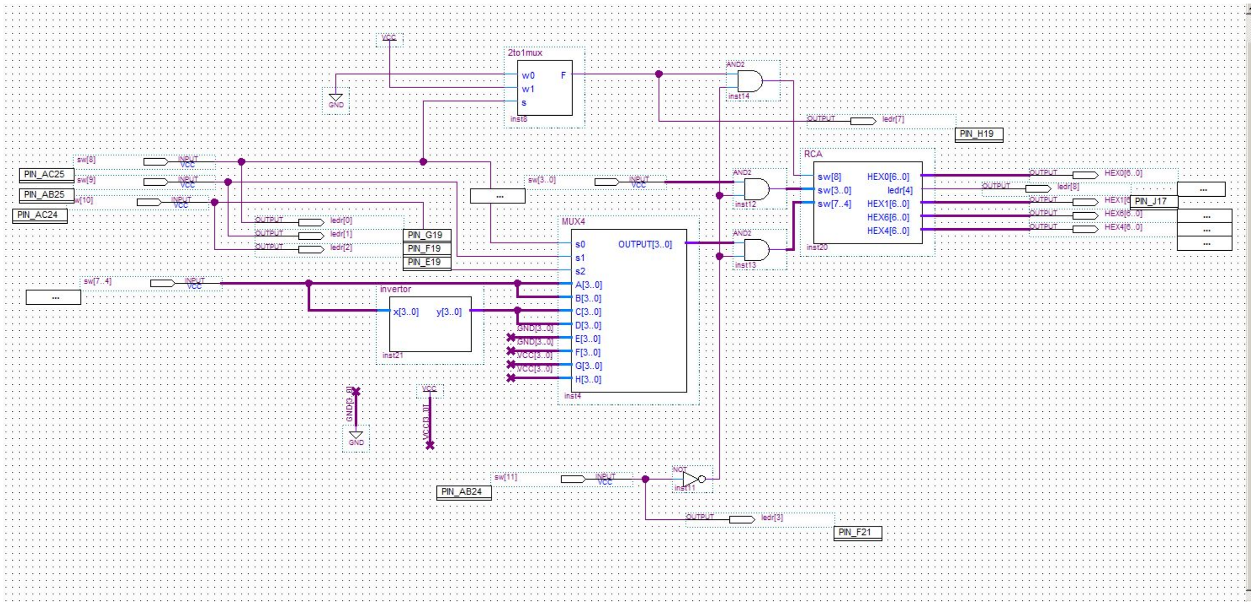
Figure 8. ALU

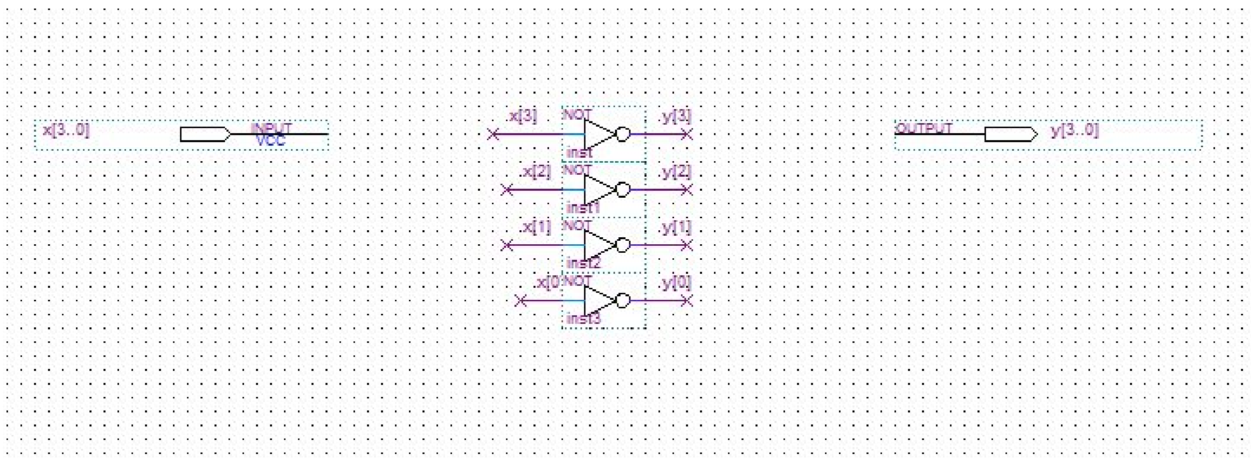

Figure 9. inverter

Testing:

5. We give a value for A and a value for B, and test each of the 8 operations in the way that we designed in prelab (as shown in Figure 10). We notice that the result will not appear to be correct if the sum is not within the range that can be represented by 4 bts, which would be an overflow condition. But within the range, the design functions correctly corresponding to the table in Figure 7.

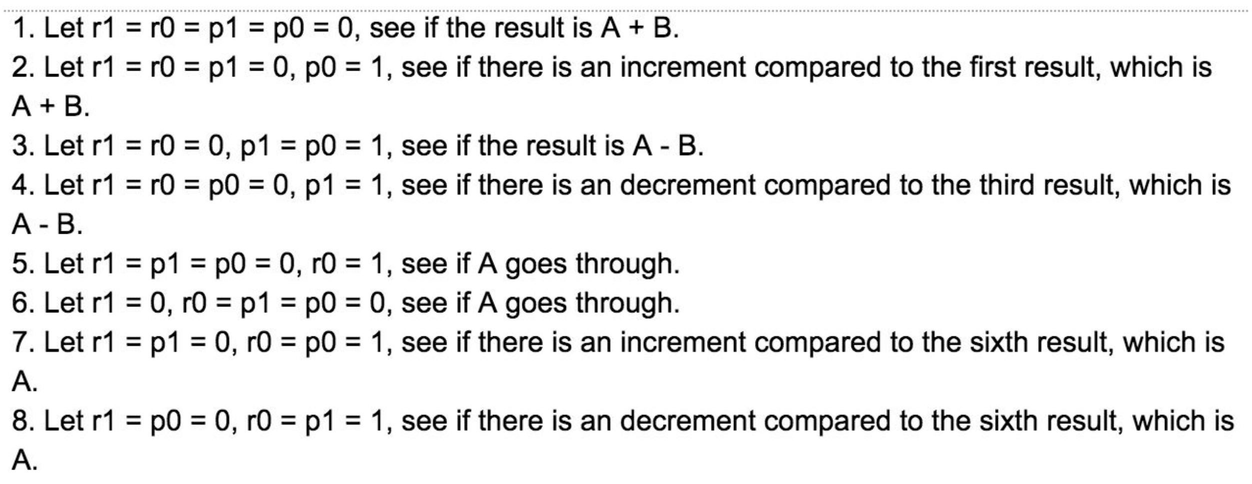
1. Let r1 = r0 = p1 = p0 = 0, see if the result is A + B.
2. Let r1 = r0 = p1 = 0, p0 = 1, see if there is an increment compared to the first result, which is A + B.
3. Let r1 = r0 = 0, p1 = p0 = 1, see if the result is A - B.
4. Let r1 = r0 = p0 = 0, p1 = 1, see if there is an decrement compared to the third result, which is A - B.
5. Let r1 = p1 = p0 = 0, r0 = 1, see if A goes through.
6. Let r1 = 0, r0 = p1 = p0 = 0, see if A goes through.
7. Let r1 = p1 = 0, r0 = p0 = 1, see if there is an increment compared to the sixth result, which is A.
8. Let r1 = p0 = 0, r0 = p1 = 1, see if there is an decrement compared to the sixth result, which is A.

Figure 10. Testing procedure

Part 2 - Add a memory save to the mini-calculator

1. Having the ALU that we just build, we add one *8dff* component to make one memory location to save the results of an arithmetic operation, and the final design is shown in Figure 11. Since we didn’t have a 4-bit output in ALU, we build a 4-bit adder (as shown in Figure 12) with carry-in which will add A[3..0] and B[3..0] and the 4-bit output will be the inputs D1 to D4 in the *8dff* component. What’s more, in order to implement the design that will choose either B[3..0] or the saved memory M[3..0], we build a MUXbm (as shown in Figure 13) which will choose B[3..0] if the switch is 1, or M[3..0] if the switch is 0.

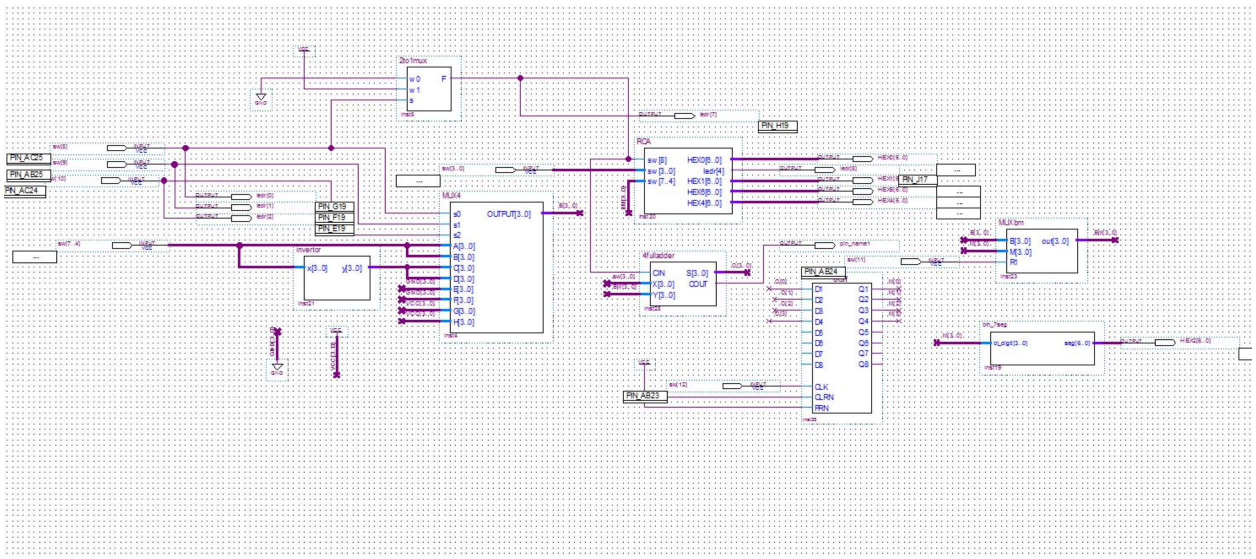

Figure 11. ALU with memory save component

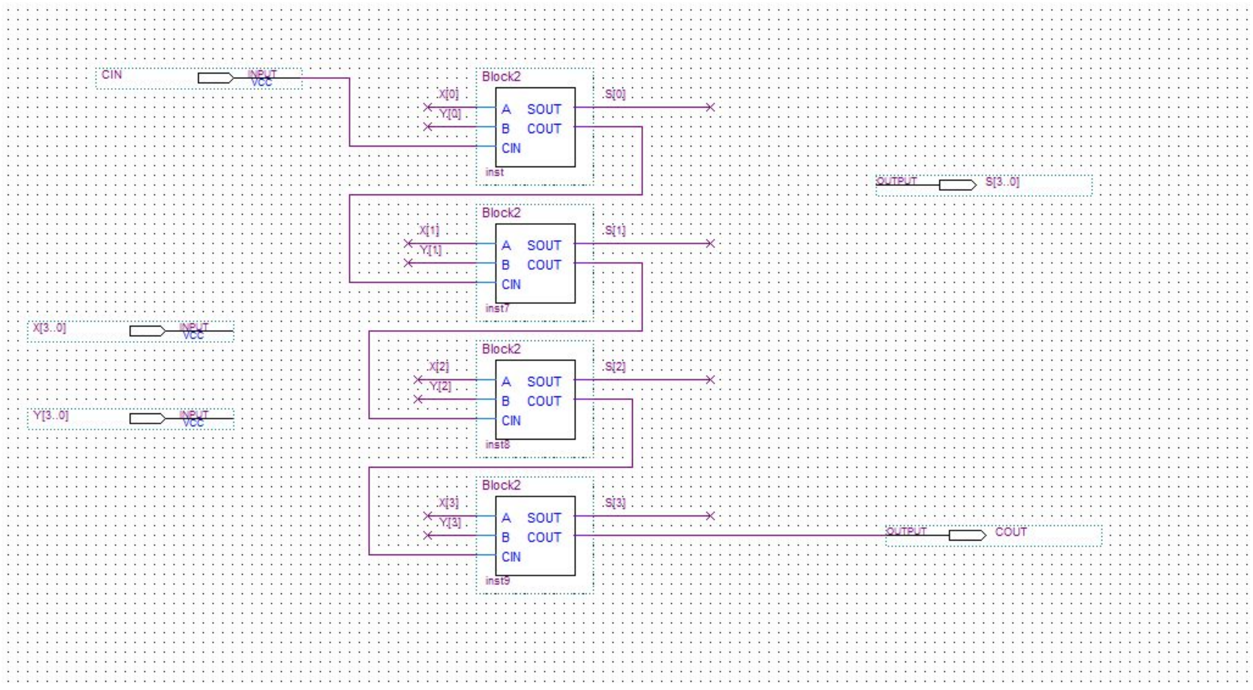

Figure 12. 4-bit adder

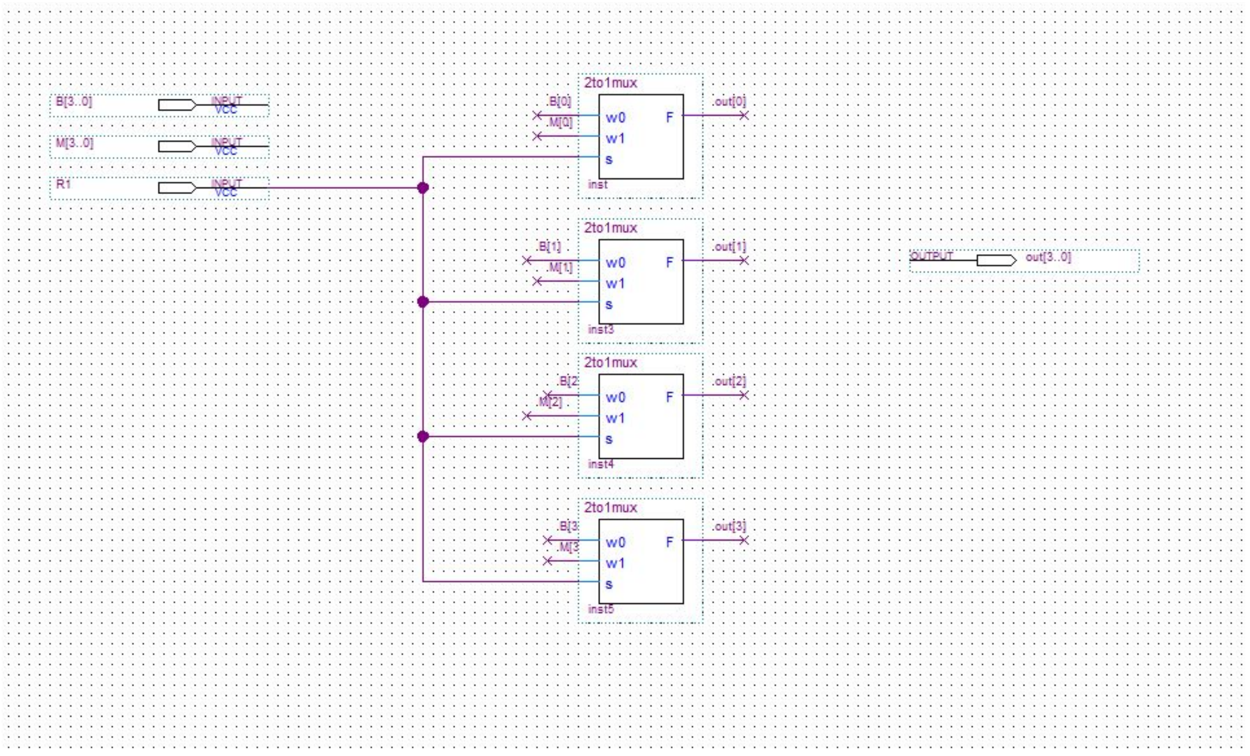

Figure 13. MUXbm

Testing

2. When r1 is 0, the ALU works as it did in Part 1.

3. When r1 is 1, we are able to save the adder output and show it from the display of M. The display of M does not change as we change the switch, and it only changes when we press the push button.
4. As demanding, we make the ALU output counts by 3s. First we make the output 3, and save the output by triggering r1. Then every time we trigger r1, the output will count by 3s, which is in the sequence 6, 9, c, and f on the 7-segment display.

IV. Mistakes that we make:

1. We can't use the output of the RCA as the input for the *8dff* component, because the width of the output is 7 bits, but we only need 4 bits for the input. So we will need a 4-bit adder which will give us the output of the addition operation.

V. Final question:

1. Negative results will appear as overflow in the seven-segment display because the circuit will not recognize the first bit of 1 as negative sign.
2. If the B input switches are 0, you can complete a transfer so that the output = A. Then you can save this value in memory and enable R1 so that it is used in place of B for the next operation. Then you could compute A - A two times (using the "A" value stored in memory) to compute A-A-A = -A. Likewise, to get an output equal to 3A, you could store A in memory and then use it compute A+A+A = 3A..
3. Use 2 4-bit ALUs to implement an 8-bit ALU. The first 4 digits of the inputs will go into the first ALU, and the carry-out of the first ALU will be the carry-in of the second ALU, whose inputs are the last 4 digits of the inputs. The result will be the sum of the sums of the 4-bit ALUs.

VI. Conclusion

The purpose of this lab was to design and test a small ALU using MUX's we had created in prior labs as building blocks. The team successfully created a 4-bit ALU with a memory operation to perform 7 basic operations: add, add and increment, subtract and decrement, subtract, transfer A, increment A and decrement A. A deeper understanding of hierarchal design and switches were developed as the team used a 8:1 and 2:1 MUX to select the operation input. We also utilized 7 segment displays and incorporated a memory unit to store the results of our ALU. Overall, the team accomplished all of the objects illustrated in the beginning of the lab.

VII. Reference lists:

- Dr. Sally Wood, Dr. Samiha Mourad, Dr. Radhika Grover. *Laboratory #6: Mini-calculator with Small Arithmetic Logic Unit(ALU).* Spring 2017. Print
- Bin_7seg_temp.txt. Spring 2017. Print
- Dr. Sally Wood, Dr. Shoba Krishnan. *7-Segment Display Tutorial.* Spring 2017. Print
- Altera Corporation - University Program. *Quartus II Introduction Using Schematic Designs.* Spring 2017. Print
- Altera Corporation - University Program. *Quartus II Introduction to Simulation of Verilog Designs.* Spring 2017. Print
- Terasic Technologies Inc. *DE2-115 User Manual.* Spring 2017. Print.