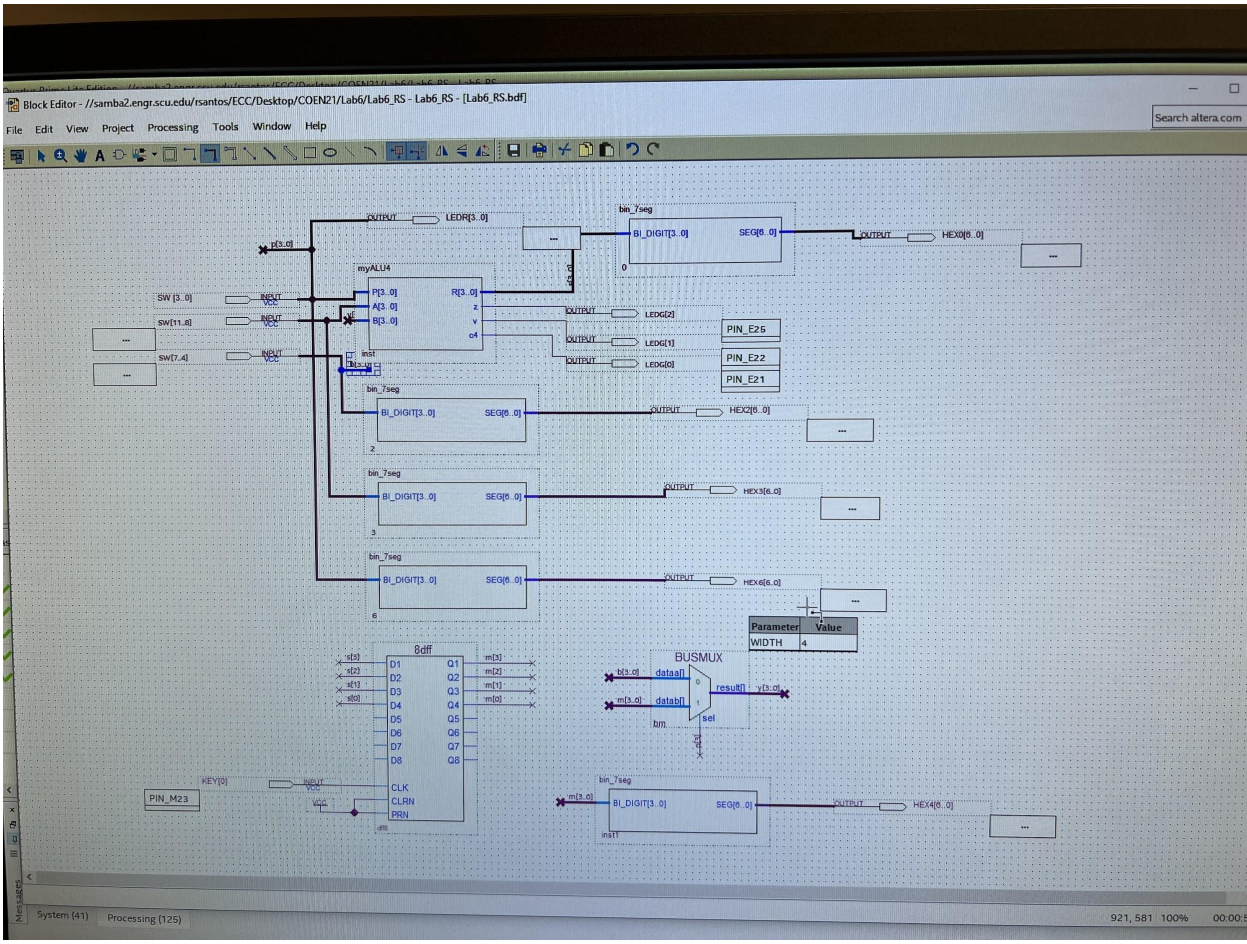
Riana Santos & Tushar Shrivastav
COEN 21 Winter 2022
Professor Ogunfunmi
18 February 2022

1. Include: Introduction, procedure, results, and conclusions
    a. Introduction
    In this lab we implemented a 4-bit Arithmetic Logic Unit that takes two 4 bit numbers and performs the following operations: (A+B), (A+B+1), (A-B-1), (A-B), (A transfer case), (A+1), (A-1), (A transfer case 2).
    We also implemented a memory case so that we can use memory like in a regular scientific calculator. That way, we can use memory values from previous calculations in the current calculations.
    b. Procedure
    Our Procedure was split into two parts. The first part asked us to implement the simple ALU we designed as part of the prelab. We implemented our verilog on the Quartus Software and added 7 segment display operations to the outputs/inputs to display them on our fpga board. In part two we followed the given instructions to add a memory module by using an 8dff module and a 4-width busmux. We tested previous operations on the board to make sure they work, and then utilized the memory module to test operations utilizing values stored in the memory.
    c. Results
    With our testing for both part 1 and 2, we performed all the operations on numbers and it worked. The save operation worked, and counting by 2's also worked.
    d. Conclusions
    In conclusion, our circuit demonstrated that it fulfilled all the requirements for the operations required, and the ability to use memory to do future calculations.
2. Include schematic, Verilog code, test plan, and results for each operation
    a. Schematic



b. Verilog Code

```
module myfulladd(a,b,cin,sout,cout);
        input a,b,cin;
        output sout,cout;

        assign sout = a ^ b ^ cin;
        assign cout = (a & b) | (b & cin) | (a & cin);
endmodule

module myadder4(X,Y,c0,S,v,c4);
        input [3:0]X,Y;
        input c0;
        output[3:0]S;
        output v,c4;
        wire [3:1]C;

        myfulladd(X[0],Y[0],c0,S[0],C[1]);
        myfulladd(X[1],Y[1],C[1],S[1],C[2]);
        myfulladd(X[2],Y[2],C[2],S[2],C[3]);
        myfulladd(X[3],Y[3],C[3],S[3],c4);
```

```
	assign v = C[3] ^ c4;
endmodule

module myALU4(P,A,B,R,z,v,c4);
	input [3:0]P,A,B;
	output [3:0]R;
	output z,v,c4;
	reg [3:0]Y;

	always @(*)
		case(P[2:1])
			2'b00: Y = B;
			2'b01: Y = ~B;
			2'b10: Y = 4'b0000;
			2'b11: Y = 4'b1111;
		endcase

	myadder4(A,Y,P[0],R,v,c4);
	assign z = ~(R[0]|R[1]|R[2]|R[3]);
endmodule


module bin_7seg(BI_DIGIT,SEG);
	input [3:0] BI_DIGIT;
	output [6:0] SEG;
	reg [6:0] SEG;
	// seg = {g,f,e,d,c,b,a};

	always @(BI_DIGIT)
		case (BI_DIGIT)
			4'h0: SEG = ~7'b0111111;	// ---a----
			4'h1: SEG = ~7'b0000110;	// |	|
			4'h2: SEG = ~7'b1011011;	// f	b
			4'h3: SEG = ~7'b1001111;	// |	|
			4'h4: SEG = ~7'b1100110;	// ---g----
			4'h5: SEG = ~7'b1101101;	// |	|
			4'h6: SEG = ~7'b1111101;	// e	c
			4'h7: SEG = ~7'b0000111;	// |	|
			4'h8: SEG = ~7'b1111111;	// ---d----
			4'h9: SEG = ~7'b1100111;
			4'ha: SEG = ~7'b1110111;
			4'hb: SEG = ~7'b1111100;
			4'hc: SEG = ~7'b1011000;
			4'hd: SEG = ~7'b1011110;
```

```
            4'he: SEG = ~7'b1111001;
            4'hf: SEG = ~7'b1110001;
        endcase
endmodule
```

c. Test Plan
   i. Set A = 3 and B = 2. Set p3 = 0.
   ii. With p2 set to 0, test the circuit with p1 and p0 both set to 0.
   iii. With p2 still set to 0, test the circuit with p1 = 0 and p0 = 1 and vice versa.
   iv. With p2 still set to 0, test the circuit with p1 and p0 both set to 1.
   v. Repeat steps 1-3 with p2 set to 1 now.
   vi. To test the save function, set A = 2 and B = 0. Now, set p3 = 1 and all other p inputs to 0.
   vii. Click KEY0 to save the number 2. The result should now read as 4 on the circuit. This will confirm that the save function works.

d. Results
   i. We encountered a few issues. One issue was that we incorrectly switched the two inputs for the BUSMUX used in our schematic, which caused our result to be incorrect. Another issue that we ran into was that the circuit was adding one more or subtracting one more than we needed. We fixed this last issue by identifying that we used A as our input to the BUSMUX instead of B as expected since A is 1 greater than B. Other than that, everything else in the lab went smoothly.

3. Describe how negative results appear in your simulation waveforms.[SIMULATION NOT NEEDED]
4. List three pairs of A, B inputs that will cause the Z output to be 1 when the operation is P = 0 0 0 0.
   a. A = F (15) B = 1
   b. A = E (14) B = 2
   c. A = 0 B = 0
5. List three pairs of A, B inputs that will cause the Z output to be 1 when the operation is P = 0 0 1 1.
   a. A = 3 B = 3
   b. A = 2 B = 2
   c. A = 1 B = 1
6. If the B input values are zero, describe a sequence of operations using memory that you could use to get a mini-calculator output of -A.

   Sequence of operations to get -A, where B = 0000, and A is any number

   - In the next state use Memory to set A = 0000 (B).
   - Transfer A to B input (B = A).
   - Select subtraction operation, and the board will give you 0 - A which is an -A output.

7. How would you connect two 4-bit ALUs to make an 8-bit ALU?
   You can connect two 4-bit ALU's to make an 8-bit ALU, by simply connecting the carry output of the first ALU to the carry input of the next 4-bit ALU.