Lab 6 – Arithmetic Logic Unit

Name: Robert Bara

Section #: 003

Date: 2/27/20

Summary/Abstract (10 points)

Lab 6 is about designing and Arithmetic Logic Unit (ALU) which will carry various binary operations between two 4bit binary inputs and display their decimal value using the cathodes on the Basys3 board.

Introduction (10 points)

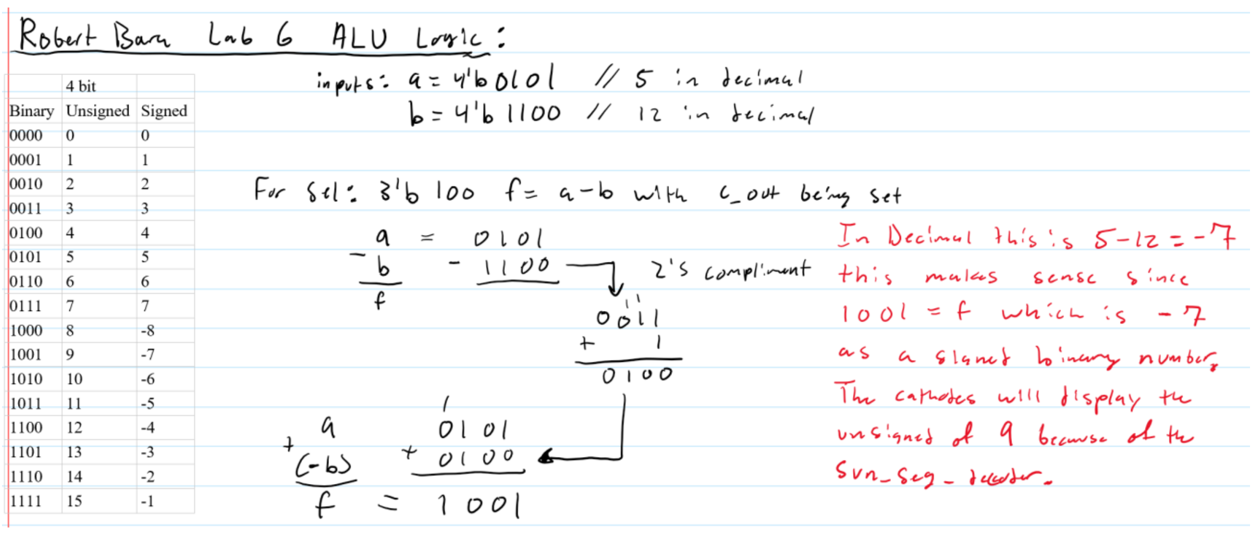
It is crucial to understand various number systems such as binary and decimal when designing an ALU because the output may display a signed or unsigned representation of the outcome of the binary operation in decimal. Verilog syntax for case statements, concatenation, and instantiation are essential for this lab. ALU's are most commonly apart of Computer Processor Units (CPU's) within computers, cellphones, and other electronics, so understanding how to program a basic ALU will help Electrical Computer Engineers understand how computer design is done.

Procedure (15 points)

To understand how this lab works, the first step is to examine the truth table provided in the lab manual for the ALU module. This truth table explains what the output f will equal based on the inputs of a, b, and the selector switches [11:9]. F is generated by using binary operations since the inputs for a and b will be binary inputs using the switches on the basys3 board. Since the output f is only 4bits, the c_out represents the carry out which will be cut off from the display that will then be projected using the cathodes from the svn_seg_decoder module. The carry in for the operation will manually be set using switch [8] based on the truth table and arithmetic. Further explanations can be seen below by calculating the correct output for the hardware implementation.

Select Operation (signal: *sel*)	**Arithmetic Operation**
3'b000	f = b with c_out = 0
3'b001	f = a + b + c_in with c_out being set
3'b010	f = a + b with c_out being set
3'b011	f = a - b - c_in with c_out being set
3'b100	f = a - b with c_out being set
3'b101	f = a + 1 with c_out being set
3'b110	f = a - 1 with c_out being set
3'b111	f = a & b with c_out = 0

[Figure 1: Truth Table for alu_1 module]



Robert Barr Lab 6 ALU Logic:

	4 bit	
Binary	Unsigned	Signed
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	-8
1001	9	-7
1010	10	-6
1011	11	-5
1100	12	-4
1101	13	-3
1110	14	-2
1111	15	-1

inputs: a = 4'b 0101 // 5 in decimal
b = 4'b 1100 // 12 in decimal

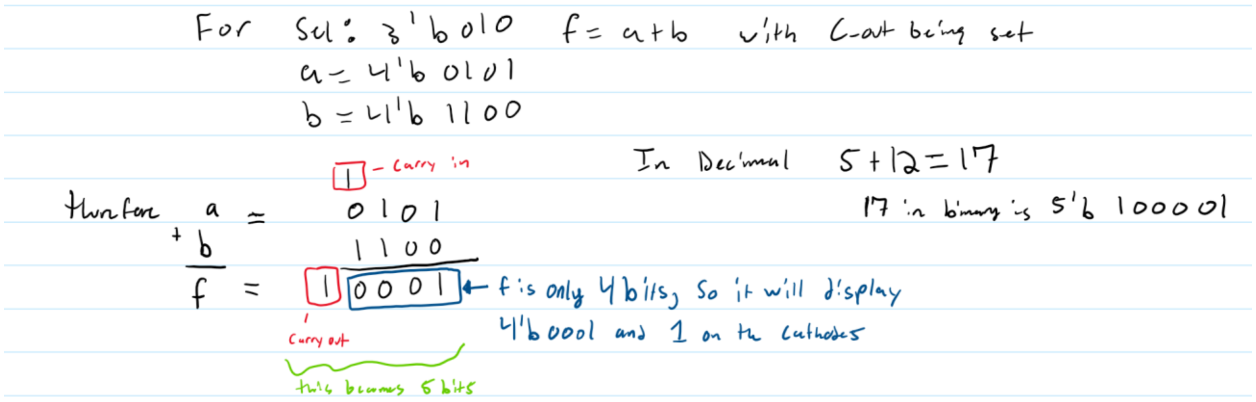
For Sel: 3'b 100 f = a − b with c_out being set

$$\begin{array}{r} a \\ -\,b \\ \hline f \end{array} = \begin{array}{r} 0101 \\ -\,1100 \\ \hline \end{array} \xrightarrow{\text{2's compliment}} \begin{array}{r} 0011 \\ +\;\;\;\;1 \\ \hline 0100 \end{array}$$

$$\begin{array}{r} a \\ +\,(-b) \\ \hline f \end{array} = \begin{array}{r} 0101 \\ +\,0100 \\ \hline 1001 \end{array}$$

In Decimal this is 5 − 12 = −7 this makes sense since 1001 = f which is −7 as a signed binary number. The cathodes will display the unsigned of 9 because of the svn_seg_decoder.

[Figure 1b: Example of case 3'b100's logic of how the ALU module calculates f]



For Sel: 3'b010 f = a + b with C-out being set
a = 4'b 0101
b = 4'b 1100

In Decimal 5 + 12 = 17
17 in binary is 5'b 10001

$$\begin{array}{rr} & \boxed{1}\ \text{- carry in} \\ \text{therefore}\ a = & 0101 \\ +\,b & 1100 \\ \hline f = & \boxed{1}\boxed{0001} \end{array}$$

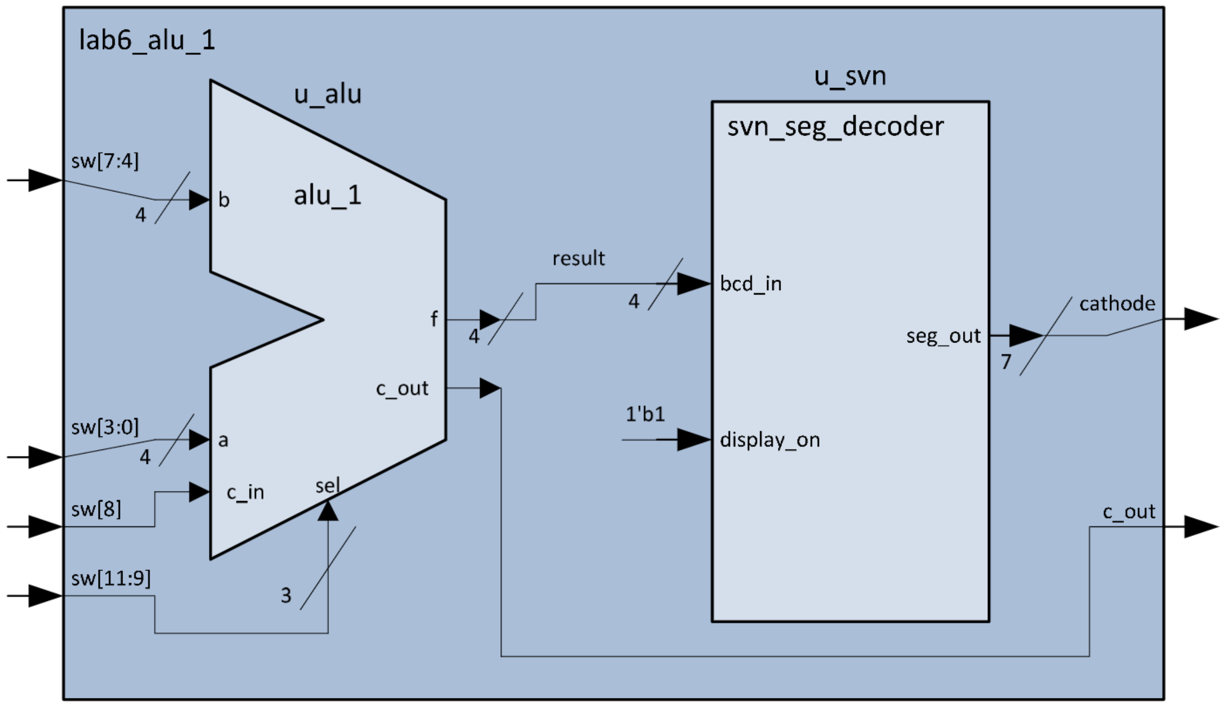
Carry out

this becomes 5 bits

← f is only 4 bits, so it will display 4'b0001 and 1 on the cathodes

[Figure 1c: Example of case 3'b010's logic of how the ALU module calculates f]

From this logic, the ALU module can be done using a case statement and concatenating {cout,f} for the output so it can be instantiated based on the block diagram below:



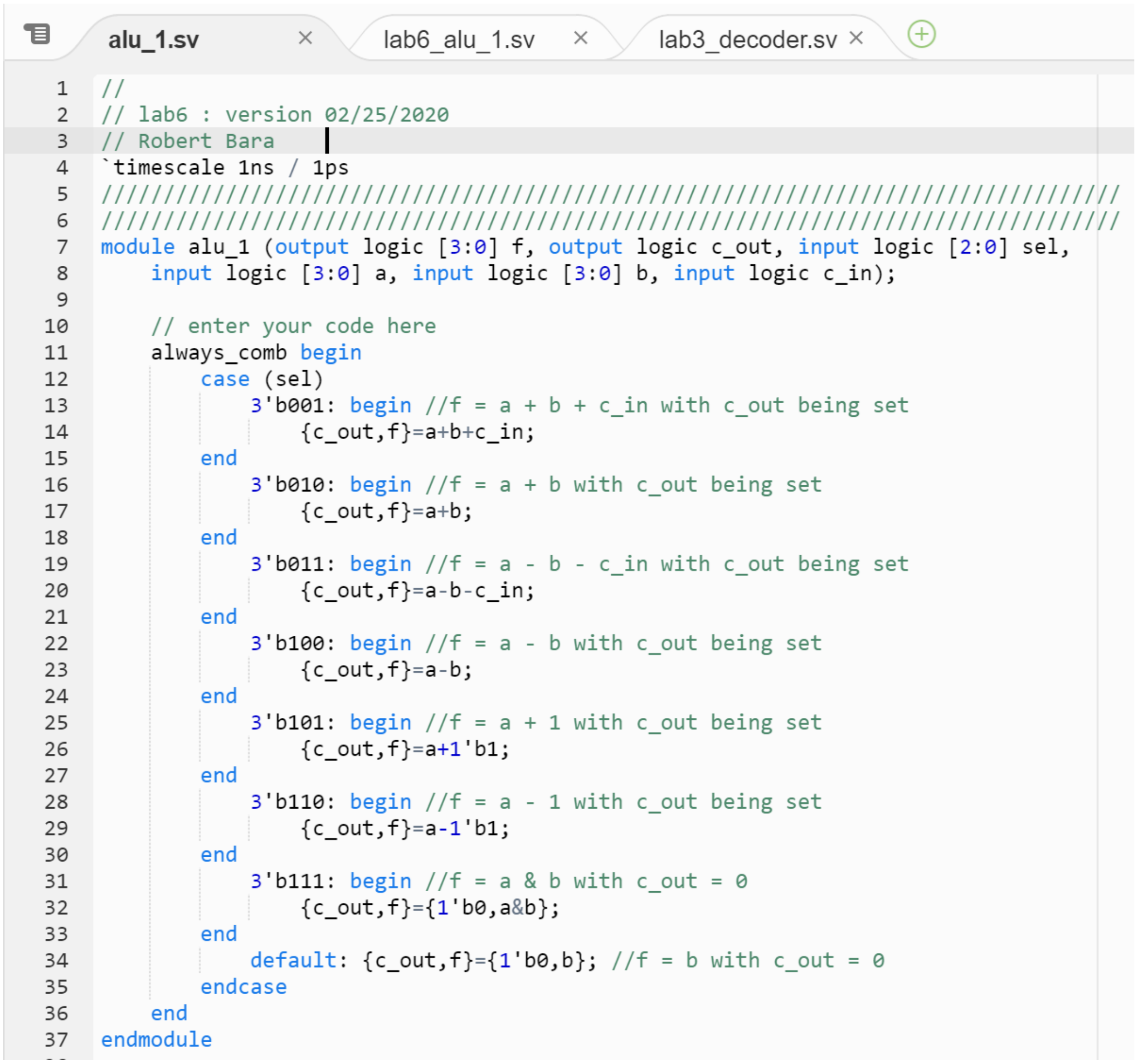
[Figure 2: Block Diagram for instantiation]

Results (45 points – see sub-sections)

Originally, I tried to complete the lab before the lab period and found the lab hard to understand but I was able to complete it within the lab period. The block diagram was a little hard to understand so I referred to how I programmed the instantiation module of lab 3 and was able to apply similar syntax to lab6 and run and simulate both modules without any mismatches. I then uploaded my .bit file to the Basys3 and examined the unassigned decimal outcome from the cathodes. As shown from the work I explained in the procedure section, I was able to verify the hardware implementation was successful and correct.

Design Code (15 points)

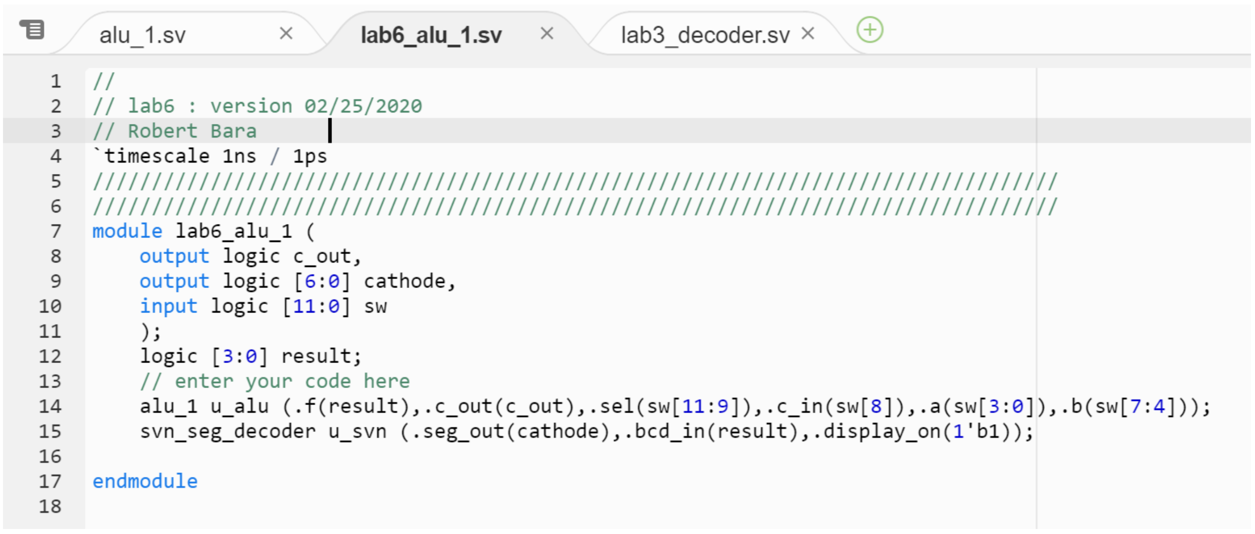
Design code for alu_1 module and the instantiation module are found below:



```
//
// lab6 : version 02/25/2020
// Robert Bara
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
module alu_1 (output logic [3:0] f, output logic c_out, input logic [2:0] sel,
    input logic [3:0] a, input logic [3:0] b, input logic c_in);

    // enter your code here
    always_comb begin
        case (sel)
            3'b001: begin //f = a + b + c_in with c_out being set
                {c_out,f}=a+b+c_in;
        end
            3'b010: begin //f = a + b with c_out being set
                {c_out,f}=a+b;
        end
            3'b011: begin //f = a - b - c_in with c_out being set
                {c_out,f}=a-b-c_in;
        end
            3'b100: begin //f = a - b with c_out being set
                {c_out,f}=a-b;
        end
            3'b101: begin //f = a + 1 with c_out being set
                {c_out,f}=a+1'b1;
        end
            3'b110: begin //f = a - 1 with c_out being set
                {c_out,f}=a-1'b1;
        end
            3'b111: begin //f = a & b with c_out = 0
                {c_out,f}={1'b0,a&b};
        end
            default: {c_out,f}={1'b0,b}; //f = b with c_out = 0
        endcase
    end
endmodule
```

[Figure 1: Code for alu_1 module]



```
//
// lab6 : version 02/25/2020
// Robert Bara
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////
module lab6_alu_1 (
    output logic c_out,
    output logic [6:0] cathode,
    input logic [11:0] sw
    );
    logic [3:0] result;
    // enter your code here
    alu_1 u_alu (.f(result),.c_out(c_out),.sel(sw[11:9]),.c_in(sw[8]),.a(sw[3:0]),.b(sw[7:4]));
    svn_seg_decoder u_svn (.seg_out(cathode),.bcd_in(result),.display_on(1'b1));

endmodule

```

[Figure 2: Code for Instantiation of alu_1 module and svn_seg_decoder from lab3]

Simulation Results (15 points)

Simulations for both modules:

```
Simulation complete - no mismatches!!!
$finish called at time : 81920 ns : File "/home/tuj22026/2613_2020s/lab6/tb_alu_1.sv" Line 67
```

[Figure 1b: Simulation for alu_1 module]

```
Simulation complete - no mismatches!!!
$finish called at time : 81920 ns : File "/home/tuj22026/2613_2020s/lab6/tb_lab6_alu_1.sv" Line 117
```

[Figure 2b: Simulation for instantiation]

Hardware Implementation (10 points)

My design was demonstrated to Sivan on the afternoon of 2/27/20 at 1:40pm during the lab period

Conclusion (10 points)

Designing an ALU is essential to software design since it is a key component for doing calculations in CPU's. Understanding the logic behind number systems such as signed and unsigned binary, as well as decimal is essential to understanding how the arithmetic logic will calculate the output. By using Verilog's case statement and concatenation I was able to create a 4'b ALU module that calculates based on two 4'b inputs a and b. I then instantiated the ALU module's output of f into the seven-segment decoder module from lab 3 to generate the 4-bit unsigned decimal output for f onto the cathodes of the Basys3 board, while c_out went directly out into the board based on the block diagram. Both of my modules simulated with no mismatches and I was able to prove that my hardware implementation is correct.