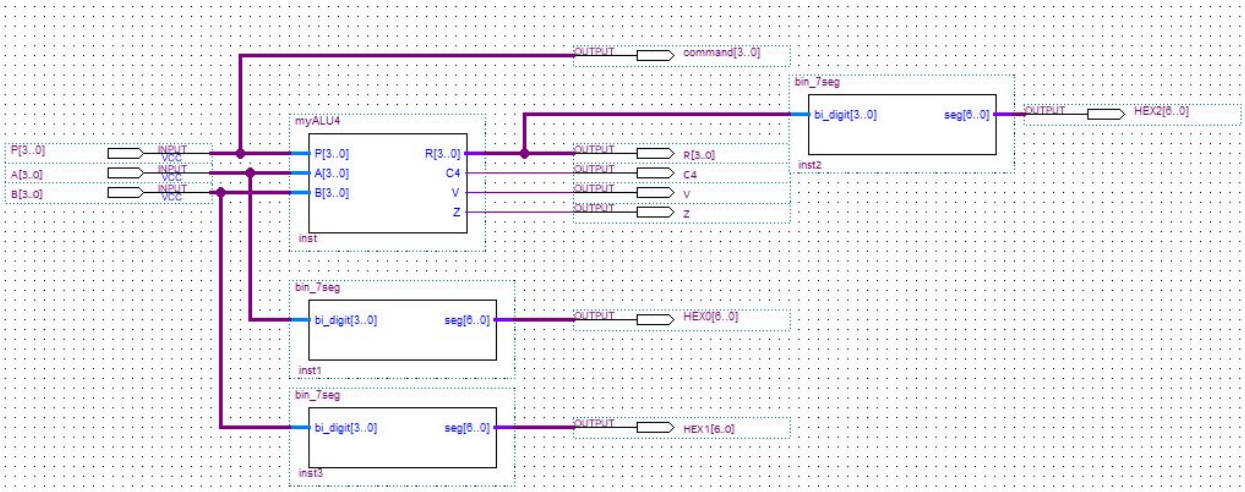
Lab Report #6
Dillon Kanai
Connor Callahan

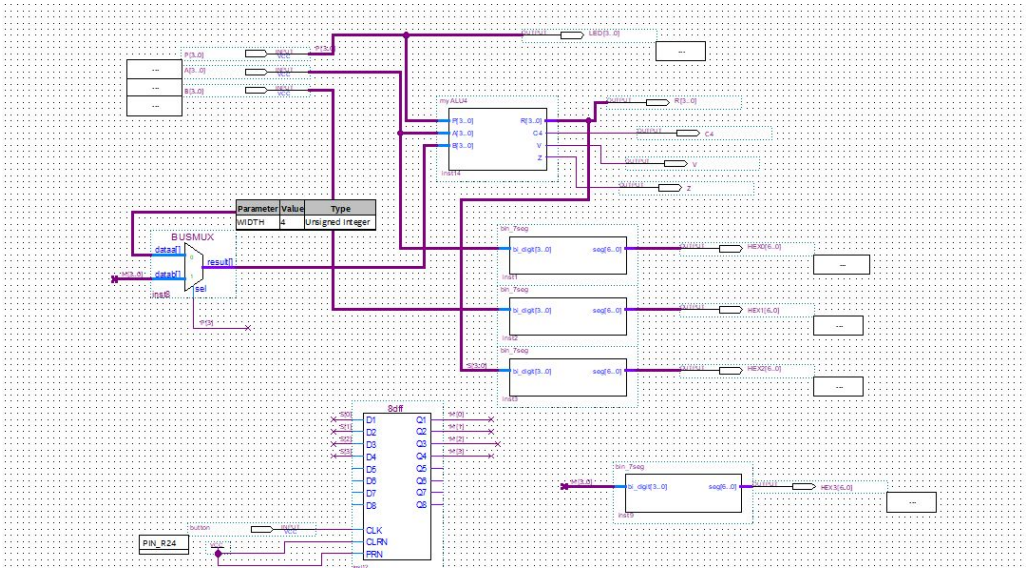
Introduction:
In this lab, we will be using two 4our bit numbers to perform a few operations. To choose each operation, we will be using the switches P0, P1, P2, P3 to select them. P0 will determine the carry in, P1 will determine whether we are subtracting B (and the carry in if applicable) from A, and P2 will determine whether B has an effect on the final sum. To perform these operations, we will use myAdder4 and myfulladds from previous labs. The rest of the variables, C4, V, and Z will hook up to LEDs and A, B, and R will hook up to 7 segment LEDs.

In part 2, we will be creating a memory function that will store the sum when a button is pushed. We will also create a switch, P3, that will substitute the value stored for B. To do this, we used a mux and p3 as the select. The inputs for this mux are B and the stored value. The button acts like a clock.

Schematic for Lab Part 1



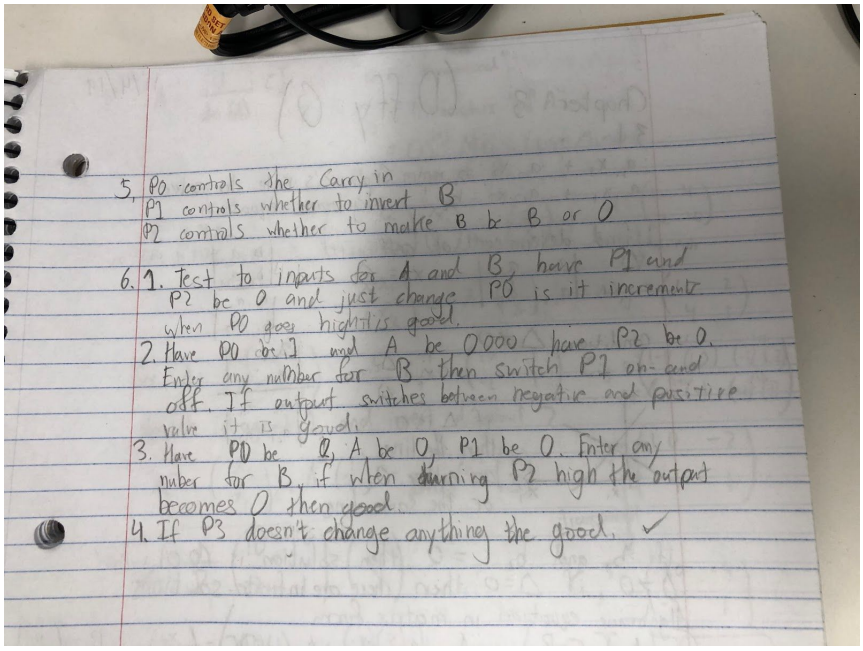
Schematic for Lab Part 2



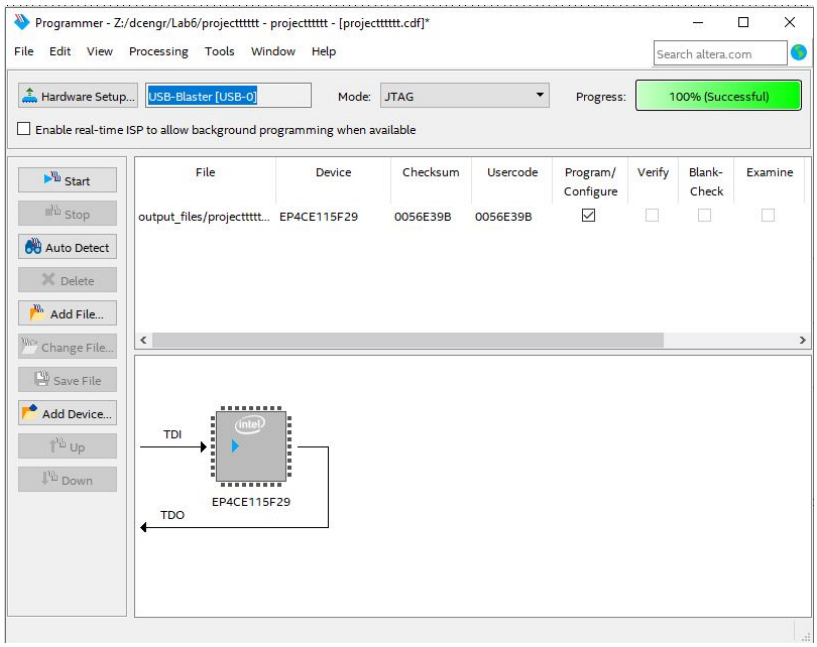
Verilog Code:

```
module mux4to1(B, D, S);
   input [3:0]B;
   input [1:0]S;
   output reg [3:0]D;

   always@(*)
      case(S)
         0: D = B;
         1: D =~B;
         2: D = 4'b0;
         3: D = 4'b1111;
      endcase
endmodule
module myfulladd(A, B, Cin, Sout, Cout);
   input A, B, Cin;
   output Sout, Cout;

   assign Sout = (A^B^Cin);
   assign Cout = (A&B)|(Cin&(A|B));

endmodule
module myadder4(X, Y, Cin, V, C4, S);
   input [3:0]X, Y;
   input Cin;
   output V, C4;
   output [3:0]S;
   wire C1, C2, C3;
   assign V = ((X[3]&Y[3]) ^ S[3]);
   myfulladd(X[0], Y[0], Cin, S[0], C1);
   myfulladd(X[1], Y[1], C1, S[1], C2);
   myfulladd(X[2], Y[2], C2, S[2], C3);
   myfulladd(X[3], Y[3], C3, S[3], C4);
endmodule
module myALU4(P, A, B, R, C4, V, Z);
   input [3:0]P, A, B;
   output [3:0]R;
   output C4, V, Z;
   wire [3:0]Y;
   assign Z = ~R[0]&~R[1]&~R[2]&~R[3];
   mux4to1 f1(B, Y, P[2:1]);
   myadder4 f2(A, Y, P[0], V, C4, R);
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;
        4'h1: seg = ~7'b0000110;            // ---a----
        4'h2: seg = ~7'b1011011;            // |      |
        4'h3: seg = ~7'b1001111;            // f      b
        4'h4: seg = ~7'b1100110;          // |      |
        4'h5: seg = ~7'b1101101;          // ---g----
        4'h6: seg = ~7'b1111101;          // |      |
        4'h7: seg = ~7'b0000111;          // e      c
        4'h8: seg = ~7'b1111111;                  // |      |
        4'h9: seg = ~7'b1101111;            // ---d----
        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'b1110101;
endcase

endmodule
```

Test Plan:



Proof of successful upload:



Responses

Describe how negative results appear in the seven segment display.

Negative results appear as 16 minus the difference between A and B inputs.

List three pairs of A, B inputs that will cause the Z output to be 1 when the operation is P = 0000 and C0 is 0

- A = 0000 and B = 0000
- A = 1111 and B = 0001
- A = 1010 and B = 0110

List three pairs of A, B inputs that will cause the Z output to be 1 when the operation is P = 0011 and C0 is 0

- A = 0001 and B = 0001
- A = 0010 and B = 0010
- A = 0011 and B = 0011

If the B input switches are zero, describe a sequence of operations using memory that you could use to get a minicalculator output of -A

You could save the result of whatever the sum of A + B (which would be A because B is 0). Then save that into memory. Then set A to 0 and then use the saved value as B by turning p3 to 1. Then turn the control signals to 0011 to subtract. The result will be -A.

How would you connect two 4-bit ALUs to make an 8-bit ALU?

To hook up two 4 bit ALUs, we need to hook up the carry out of the first ALU to the second ALU. Instead of having two V outputs, we only need to have the V output come from the second ALU. For the Z value, we need to check if all 8 bits are one. So we would have a statement twice as long than the one in this lab.