II. PRE-LAB

1. Be sure you understand 2's complement representation of signed numbers. See Appendix A. The 4-bit adder can be used for adding or subtracting since (-B) represented in 2's complement form can be added to A to get A-B = A+(-B).

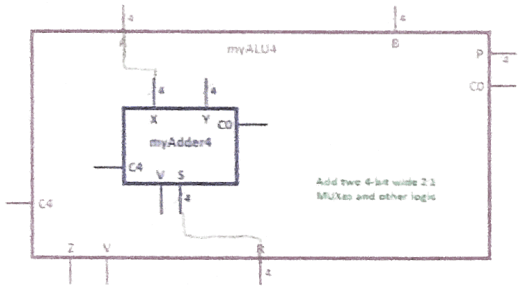
   The bit operations for addition are the **same for both** unsigned addition and signed addition using 2's complement representation. This is why 2's complement representation is so widely used. The bit pattern result is the same for both, but the **interpretation of the value of the bits will be different for signed and unsigned numbers**. (See Appendix A).

   Fill in the table below for your adder **Y** inputs and carry-in input for each of the four operations listed. The operations are specified by **p1** and **p0** when **p3** = **p2** = 0. The first entry has already been completed for you.

Op-Select Input	**Adder 4-bit X inputs**	**Adder 4-bit Y inputs**	**Adder Carry-in**	**Operation**	**Notes**
p3 p2 p1 p0	**a3 a2 a1 a0**	**b3 b2 b1 b0**			
0 0 0 0	a3 a2 a1 a0	b3 b2 b1 b0	0	R=A+B	Add
0 0 0 1	a3 a2 a1 a0	b3 b2 b1 b0	1	R=A+B+1	Add and Increment
0 0 1 0	a3 a2 a1 a0	b3'b2'b1'b0'	0	R=A-B-1	Subtract and Decrement
0 0 1 1	a3 a2 a1 a0	b3'b2'b1'b0'	1	R=A-B	Subtract

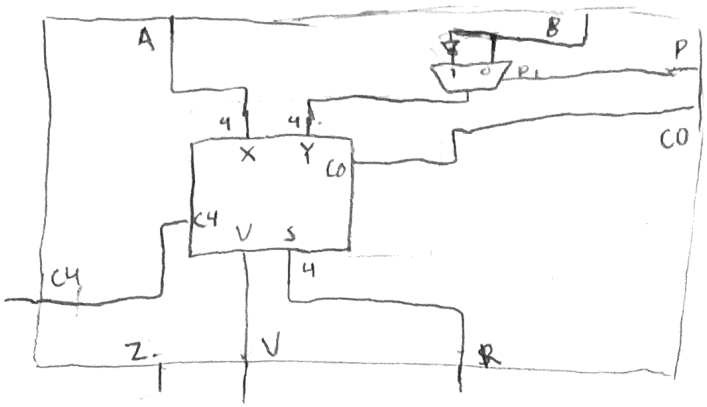
   The *myadder4* component from the previous lab (Lab 5) will be used to perform the ALU's addition. The ALU **A** input will be connected to the adder **X** input, and the adder **S** output will be the ALU **R** output. The adder **Y** input and carry-in to the adder will depend on which operation is selected, as defined by the table above.

2. Use a 4-bit adder symbol, a 4-bit wide 2:1 MUX, and other logic components as needed to design this first part of the arithmetic unit. Based on your table, use a 4-bit wide 2:1 MUX to produce the **Y** adder input with **p1** as the select. Add logic for the adder carry in input. Draw the schematic for your design.

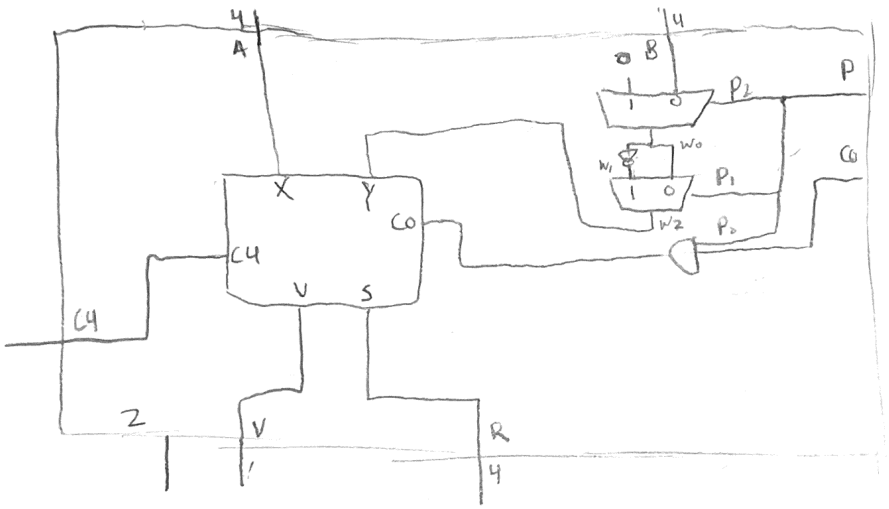


3. Modify your design to include the next four operations shown in the problem statement table when **p2**=1. A second 4-bit wide 2:1 MUX with **p2** as the select can be used to create the inputs to the MUX used in the previous step. Draw a schematic for your design.
4. Write a hierarchical Verilog module *myALU4* which uses *myadder4*. The ALU inputs will be P, A, B, and C0. The outputs will be R, C4, V, and Z. It will perform the eight operations described in the problem statement.
   a. Add logic to make the carry input to the adder.

2)



3)



4)

```
module   myALU4 (A, B, P, CO, C4, Z, V, R);
    input   [3:0] A, B, P;
    input   CO;
    output  [3:0] R;
    output  C4, Z, V;
    wire    [4:0] W;
    assign W3 = P[0] & CO;
    assign W2 = (B Mux functions)
    myAdder4 (A, W2, W3, R, C4, V);
    assign Z = ~R[0] & ~R[1] & ~R[2] & ~R[3];
endmodule;
```

Since we don't know how to do 4c, I'm going to call the output of the two muxes (input to Y) wire 2.

B → W2.

5) - p0 controls whether there is a carry in or not for for the adder.

- p1 controls whether something is being added to or subtracted from A. So it controls the operation (add/subtract) being performed on A.
- p2 controls the B input. If p2 is 0, then B is being added or subtracted from the equation. If p2 is 1, then B input is not a part of the operation being done to A input.

6) Step 1: Set P=0000, A=0010, B=0001, C0=0

Outputs R=0011, Z=0, V=0, C4=0

- Test that the two Muxes are working, no carry in and that p operations are right

Step 2: Set P=0001, A=0010, B=0001, C0=1

Outputs R=0100, Z=0, V=0, C4=0

- Test: Addition with increment and that carry in is working

Step 3: Set P=0010, A=0010, B=0001, C0=0

Outputs R=0000, Z=1, V=0, C4=0

- Test subtraction with decrement and that Z is working

Step 4: Set P=0011, A=0010, B=0001, C0=1

Outputs R=0001, Z=0, V=0, C4=0

- Test subtraction and that carry in of 1 is being done.