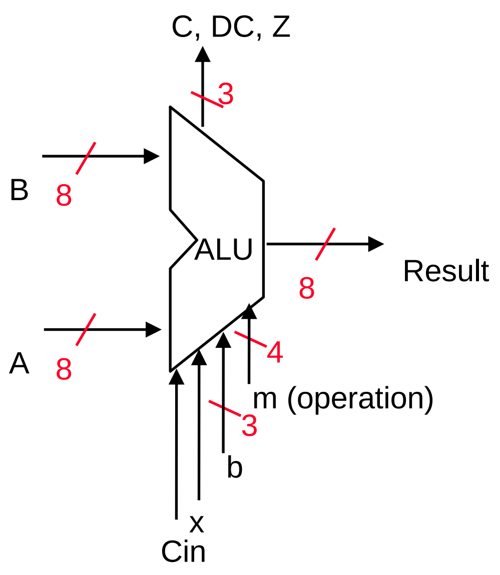
University of Jordan
Computer Engineering Department
CPE439: Computer Design Lab

Experiment 3: 8-Bit ALU

It is required to construct and test a Verilog module for an 8-bit arithmetic and logic unit (ALU) suitable for incorporation in your PIC16F84A design. This ALU should use the adder/subtractor you have built in Experiment 2. You should use modular design where you start by building and testing low-level modules using the library modules defined in **Lib439.v**, then use the low-level modules in larger modules. The symbol and function of this ALU are shown below. Where "C" is the Carry/Borrow' bit, "DC" is the Digit Carry/Borrow' bit, "Z" is the Zero bit, "Cin" is the carry in bit used in the rotate operations, "x" is an input bit, and "b" is the bit select field.



m	Function	
0000	B + A	Add
0001	B + 1	Increment
0010	B – A	Subtract
0011	B – 1	Decrement
0100	A • B	And
0101	A + B	Or
0110	A + B	Xor
0111	$\overline{B}$	Complement
1000	B	Move B
1001	$\overleftarrow{B}$	Rotate left
1010	$\overrightarrow{B}$	Rotate right
1011	$\widehat{B}$	Swap nibbles
1100	0	Clear
1101	A	Move A
1110	B<b> ← x	**Bit modify**: x should replace bit B<b> and the modified B should be copied to the output.
1111	B<b> == x	**Bit test**: Set Z if bit B<b> equals x.

We suggest that you build this unit using four smaller modules: arithmetic, logic, rotate, and bit module.

Multiplexer

Since there are 4 modules each has four functions, we suggest that you build a 4-to-1 multiplexer module that has Verilog code similar to the following code:

```
module Mux_4_to_1(Output, s, i0, i1, i2, i3);
  output Output;
  input [1,0] s;
  input i0, i1, i2, i3;
  // implementation details are left to the student
  …
endmodule
```

This module should be used to build an 8-bit multiplexer similar to the following code:

```
module Mux_4_to_1_8b(Output, s, i0, i1, i2, i3);
  output [7:0] Output;
  input [1,0] s;
  input [7:0] i0, i1, i2, i3;
  // implementation details are left to the student
  …
endmodule
```

Decoder

To implement the bit modify and test operations, we suggest that you build a 3-to-8 decoder that is similar to the following code:

```
module Decoder_3_to_8(Output, Input);
  output [7,0] Output;
  input [2,0] Input;
  // implementation details are left to the student
  …
endmodule
```

8-Bit ALU

After building the logic, rotate, and bit modules, assemble the four modules using a 4-to-1 multiplexer. Your ALU module should have Verilog code similar to the following code:

```
module ALU(Result, C, DC, Z, A, B, m, b, Cin, x);
  output [7:0] Result;
  output C, DC, Z;
  input [7:0] A, B;
  input [3:0] m;
  input [2:0] b;
  input Cin, x;
  // implementation details are left to the student
  …
endmodule
```

Report

Your report should include detailed design, Verilog code for all modules including your test modules, and timing diagram that demonstrates the correct operation of your design given the input shown in the following table. Also estimate the maximum delay expected for you design.

A	B	m	b	Cin	x
0101 1001	0011 1001	0000	000	0	0
0101 1001	0111 1111	0001	000	0	0
0101 1001	0011 1001	0010	000	0	0
0101 1001	1000 0000	0011	000	0	0
0101 1001	0111 1111	0100	000	0	0
0101 1001	0111 1111	0101	000	0	0
0101 1001	0111 1111	0110	000	0	0
0101 1001	0111 1111	0111	000	0	0
0101 1001	0111 1111	1000	000	0	0
0101 1001	0011 1001	1001	000	1	0
0101 1001	0011 1001	1010	000	1	0
0101 1001	0011 1001	1011	000	0	0
0101 1001	0011 1001	1100	000	0	0
0101 1001	0011 1001	1101	000	0	0
0101 1001	0011 1001	1110	010	0	1
0101 1001	0011 1001	1111	010	0	1