Ryan Sullivan U73687595

EC311 Lab1

**Half Adder:**

module half\_adder(

input A,

input B,

output S,

output C

);

assign S = A^B;

assign C = A&B;

endmodule

**Full Adder:**

module full\_adder(

input A,

input B,

input Cin,

output S,

output Cout

);

wire s1, s2, c1, c2;

half\_adder h\_a(

.A(A),

.B(B),

.S(s1),

.C(c1)

);

half\_adder h\_b(

.A(s1),

.B(Cin),

.S(s2),

.C(c2)

);

assign S = s2;

assign Cout = c1|c2;

endmodule

**Four Bit Full Adder-Subtractor:**

module fourbit\_addsub(

input [3:0] A,

input [3:0] B,

input M,

output [3:0] S,

output Cout,

output V

);

wire s0,s1,s2,s3,c0,c1,c2,c3;

full\_adder fa\_a(

.A(A[0]),

.B(B[0]^M),

.Cin(M),

.Cout(c0),

.S(s0)

);

full\_adder fa\_b(

.A(A[1]),

.B(B[1]^M),

.Cin(c0),

.Cout(c1),

.S(s1)

);

full\_adder fa\_c(

.A(A[2]),

.B(B[2]^M),

.Cin(c1),

.Cout(c2),

.S(s2)

);

full\_adder fa\_d(

.A(A[3]),

.B(B[3]^M),

.Cin(c2),

.Cout(c3),

.S(s3)

);

assign S = {s3, s2, s1, s0};

assign Cout = c3;

assign V = c2^c3;

endmodule

**ALU Adder Module:**

module fbit\_add(

input [3:0]A,

input [3:0]B,

output [7:0]Y

);

assign Y = A+B;

endmodule

**ALU Multiplier Module:**

module fbit\_mult(

input [3:0]A,

input [3:0]B,

input [7:0]Y

);

assign Y = A\*B;

endmodule

**ALU Concatenator Module:**

module fbit\_concat(

input [3:0]A,

input [3:0]B,

output [7:0]Y

);

assign Y = {A,B};

endmodule

**ALU Left-Shift Module:**

module fbit\_lshift(

input [3:0]A,

input [3:0]B,

output [7:0]Y

);

assign Y = A<<B;

endmodule

**ALU Multiplexer:**

module mux(

input [1:0]S,

input [7:0]A,

input [7:0]B,

input [7:0]C,

input [7:0]D,

output [7:0]Y

);

reg [7:0] y;

always @(S,A,B,C,D) begin

case(S)

2'b11: y <= B;

2'b10: y <= C;

2'b01: y <= A;

2'b00: y <= D;

endcase

end

assign Y = y;

endmodule

**ALU Module:**

module alu(

input [3:0]A,

input [3:0]B,

input [1:0]S,

output [7:0]Y

);

wire [7:0]a;

wire [7:0]b;

wire [7:0]c;

wire [7:0]d;

wire [7:0]y;

fbit\_add add(

.A(A),

.B(B),

.Y(a)

);

fbit\_mult mult(

.A(A),

.B(B),

.Y(b)

);

fbit\_lshift ls(

.A(A),

.B(B),

.Y(c)

);

fbit\_concat con(

.A(A),

.B(B),

.Y(d)

);

mux m(

.A(a),

.B(b),

.C(c),

.D(d),

.S(S),

.Y(y)

);

assign Y = y;

endmodule