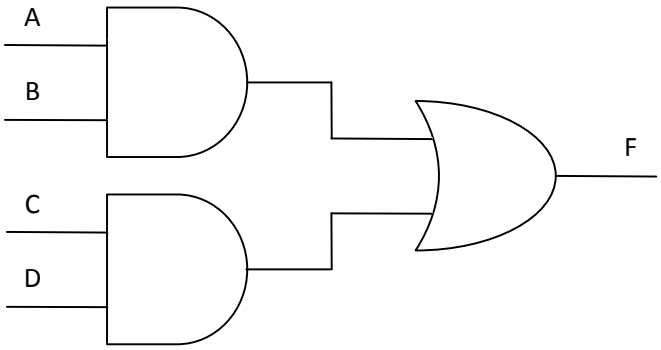
February 13th, 2012 **2612 – Digital Design – Exam 1**

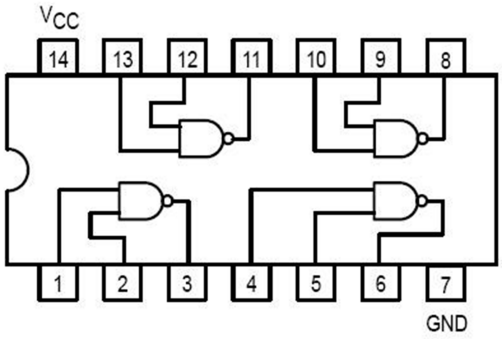
Name_____

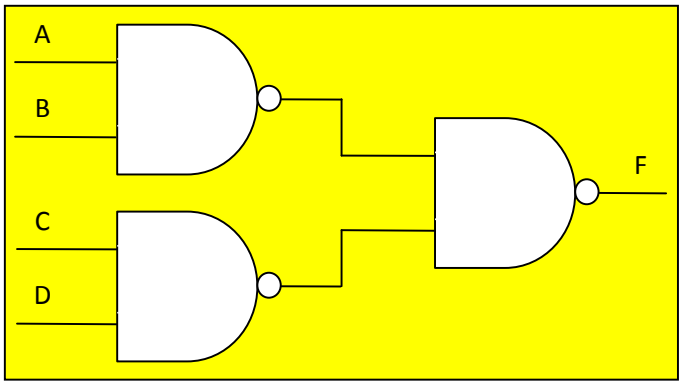
1. [10 points]
   a. Write the Verilog equation for the circuit below using the ***assign*** statement.




```
assign F = (A & B) | (C & D);
```

   b. You are given the 7400 device shown below (quad NAND gates) to implement this circuit. Draw an equivalent schematic using a minimum number of only NAND gates.





2. [10 points] Binary number conversions/representations
   a. Convert the following binary number to decimal: 8'b10000011

      131

   b. Convert the following binary number to hexadecimal: 12'b111010100101

      EA5

   c. You need to design a decoder to drive a 5x7 LED display. The number of output signals is obviously 35. What is the minimum number of bits you need for the input?

      6

   d. The 2's complement is a way to represent negative integers or binary numbers. The 2's complement is complement of the absolute value of the number plus 1. Convert: -5 into an 8 bit binary 2's complement number.

      5 binary -> 00000101 complement -> 11111010 plus 1 -> 11111011

   e. Convert +10 into an 8 bit binary number. Add to this the binary number (-5) from (d) above to demonstrate that 10 + (-5) = 5.

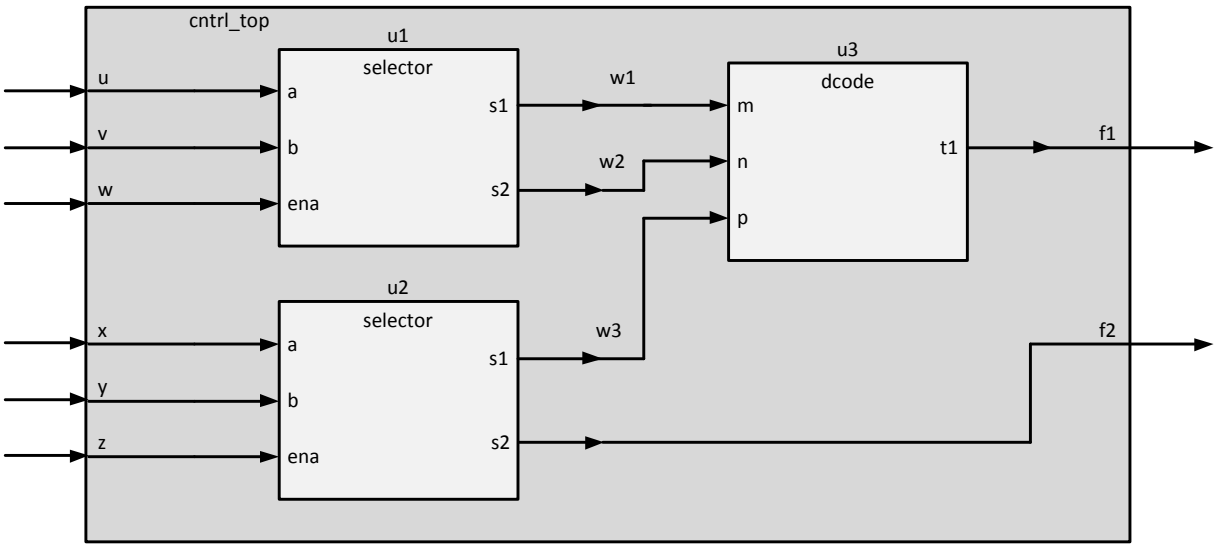
```
(carries) -> 1111 1
10 binary -> 00001010
(-5) ->      11111011
sum ->       00000101  = 5
```

3. [15 points]For the Verilog module described below, fill in the truth table.

```
module fs (a, b, cin, f, cout);
     input a, b, cin; output f, cout; reg f, cout;
     always @(a, b, cin) begin
          if (cin == 1'b0) begin
               f = a ^ b;
               cout = ~a & b;
          end else begin
               f = ~(a ^ b);
               cout = ~(a & ~b);
          end
     end
endmodule
```

cin	b	a	f	cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	1
1	1	1	1	1

4. [20 points]The block diagram below illustrates a structural module that contains three instances of two modules. Write the Verilog code for module **cntrl_top** that implements this design.



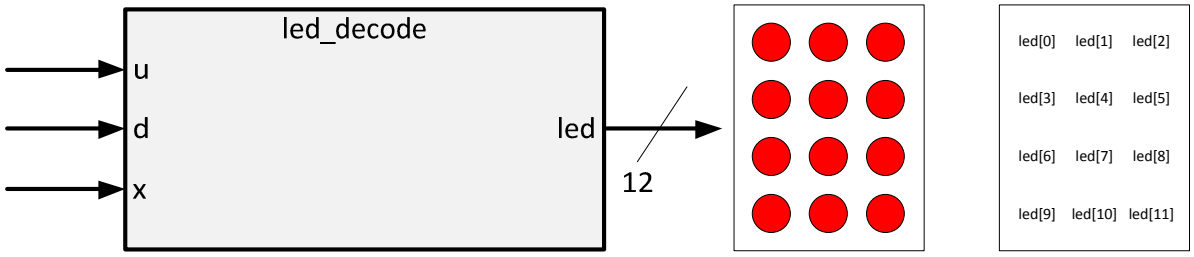

```
module cntrl_top (u, v, w, x, y, z, f1, f2);
     input u, v, w, x, y, z;
     output f1, f2;

     wire w1, w2, w3;

     selector u1 (.a(u), .b(v), .ena(w), .s1(w1), .s2(w2));
     selector u2 (.a(x), .b(y), .ena(z), .s1(w3), .s2(f2));
     dcode u3 (.m(w1), .n(w2), .p(w3), .t1(f1));

endmodule
```

5. [20 points] The following is a truth table for a 3 bit to 8 bit one-hot decoder. Create the Verilog design based on the skeleton provided below.

ena	b_in			d_out							
	[2]	[1]	[0]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

```
module onehot8 (ena, b_in, d_out);
     input ena; input [2:0] b_in; output [7:0] d_out;
     reg [7:0] d_out;

     always @(ena, b_in) begin
          case(b_in)
               3'b000: d_out = 8'b00000001;
               3'b001: d_out = 8'b00000010;
               3'b010: d_out = 8'b00000100;
               3'b011: d_out = 8'b00001000;
               3'b100: d_out = 8'b00010000;
               3'b101: d_out = 8'b00100000;
               3'b110: d_out = 8'b01000000;
               default: d_out = 8'b10000000;



          endcase   // end of case

          if (ena == 0) d_out = 8'b00000000;

     end  // end of always



endmodule
```

6. [25 points] An elevator display component is made up of a 3x4 LED matrix illustrated below. Design a decoder to drive it that has 3 inputs and 12 outputs. The 3 inputs (active high) are up (u), down (d) and out of service (x). The decoded outputs drive (active high) the 12 LED's to make an up arrow, down arrow or 'X' pattern on the array. Normally only one input signal (u, d or x) will be active at any time. However, if more than one input signal is active, the priority will be x, d then u.



a) Fill in the truth table for this design.

u	d	x	led											
			[11]	[10]	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x	x	1	0	0	0	1	0	1	0	1	0	1	0	1
x	1	0	0	1	0	1	1	1	0	1	0	0	1	0
1	0	0	0	1	0	0	1	0	1	1	1	0	1	0

b) A skeleton of the Verilog module is shown below. Incorporate the design using either sum of products with assign statements or conditional statements with the always construct – your choice.

```
module led_decode (u, d, x, led);
     input u, d, x;
     output [11:0] led;
     reg [11:0] led;

     always @(u,d,x) begin
          // default is everything 0
          led = 12'b000000000000;
          // next lowest priority is u
          if (u == 1) led = 12'b010010111010;
          // next priority is d
          if (d == 1) led = 12'b010111010010;
          // highest priority is x
          if (x == 1) led = 12'b000101010101;
     end

endmodule
```