Prelab Questions

Question 1: Verilog code with comments for the 2:4 binary decoder, the 4:2 binary encoder, and the 4:2 priority encoder. Do not use behavioral Verilog for these descriptions! Use the structural and dataflow concepts introduced in the previous lab.

Code Block 1: 2:4 Binary Decoder

```
1  `timescale 1ns / 1ps
2  `default_nettype none
3  //////////////////////////////////////////////////////
4  // Module Name: 2_4_binary_decoder
5  // Author: Joseph M Martinsen
6  //
7  //////////////////////////////////////////////////////
8  module two_four_decoder( Y, W, En);
9      // Declare output wires
10     output wire [3:0] Y
11
12     // Declare input wires
13     input wire [1:0] W,
14     input wire En,
15
16     // Assign output wire logic values
17     assign Y[0] =(En & ~W[1] & ~W[0]);
18     assign Y[1] =(En & ~W[1] & W[0]);
19     assign Y[2] =(En & W[1] & ~W[0]);
20     assign Y[3] =(En & W[1] & W[0]);
21 endmodule
```

Code Block 2: 4:2 Binary Encoder

```
1  `timescale 1ns / 1ps
2  `default_nettype none
```

```
3  //////////////////////////////////////////////////////////////////
4  // Module Name: 4_2_binary_encoder
5  // Author: Joseph M Martinsen
6  //
7  //////////////////////////////////////////////////////////////////
8  module four_two_encoder (W,Y,zero);
9    // Declare output wires
10   output wire [1:0] Y;
11   output wire zero;
12   //Declare input wires
13   input wire [3:0] W ;
14   // Assign logic values for output wires
15   assign Y[0] = W[1] | W[3];
16   assign Y[1] = W[2] & W[3];
17   assign zero = ~W[0] & ~W[1] & ~W[2] & ~W[3];
18 endmodule
```

Code Block 3: 4:2 Priority Binary Encoder

```
1  module priority_encoder(
2    input wire [3:0] W,
3    output wire [1:0 ] Y,
4    output wire zero)
5
6    wire [3:0] i;
7
8    assign i[0] = ~W[3] & ~ W[2] & ~ W[1] & W[0];
9    assign i[1] = ~W[3] & ~ W[2] & W[1];
10   assign i[2] = ~W[3] & W[2];
11   assign i[3] = W[3];
12
13   four_two_encoder 4_2_enc (i,Y,zero);
```

```
14
15 endmodule
```

Question 2: The complete truth table for the gate-level schematic shown in Figure 2. This truth table should not include dont cares (i.e. X)!

w_0	w_1	w_2	w_3	y_1	y_0	$zero$
0	0	0	0	0	0	1
0	0	0	1	1	1	1
0	0	1	0	1	0	1
0	0	1	1	1	1	1
0	1	0	0	0	1	1
0	1	0	1	1	1	1
0	1	1	0	1	1	1
0	1	1	1	1	1	1
1	0	0	0	0	0	1
1	0	0	1	1	1	1
1	0	1	0	1	0	1
1	0	1	1	1	1	1
1	1	0	0	0	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	0

Question 3: A brief comparison of the the behavioral implementation of a multiplexer described in the background section with the multiplexer you described in the previous lab using structural and dataflow.

First off, in the behavioral implementation, there are no ouput wires. Instead there is a output reg, unique to an always block. The big difference is that in the structual code, the gate levels must be instatiated. Compare that to the behavioral which has an always block with a nested if else statement. The if else statement drives the logic instead of having to use logic gates.