George Suarez
CSE 310

Homework 5

1. What are the four possible output values of a 1-bit binary variable in Verilog?

   - Logic 0
   - Logic 1
   - Unknown Logic Number x
   - Higher Impendence Number z

2. Which of the statements about Verilog is/are true?

   - In Verilog, one module can instantiate other modules, and can have multiple instances of another module.
   - *always* @ * can be used in describing Combinational logic.
   - *always* @ * can be used in describing memory made of flip-flops and registers.
   - *always* @ * can be used preceding a *case* statement.

3. 2x1 Mux Module

```
module mux_2to1( input wire din_0, input wire din_1, input wire S,

         input wire E, output wire out);

  wire and1, and2, not1;

  assign not1 = !S;
  assign and1 = n1 & din_0 & E;
  assign and2 = din_1 & S & E;
  assign out = a1 | a2;

endmodule
```

4x1 Mux Module

```
module mux_4to1( input wire [3:0] I, input wire [1:0] S, input wire E,

                 output wire y);

  //put internal wires here
  wire w1, w2;

  //create instances of mux_2to1 for connections
  mux_2to1 ( w1, I[0], I[1], S[0], E );
  mux_2to1 ( w2, I[2], I[3], S[1], E );
  mux_2to1 ( y, w1, w2, S[1], E );

endmodule //End Of Module mux_4to1
```

4. The following Verilog code is a behavioral model of a 4-to-1-line multiplexer and the stimulus. Fill in the missing code underlined.

```
//Design module of 4-to-1-line MUX

module mux4_1_bh (I, select, y); //I is data input, select is selection
lines, y is output

input [3:0] I;

input [1:0] select;

output reg y; //y is of data type reg

always @ ( select[0] or select[1])

case ( select )

2'b00: y = I[0];

2'b01: y = I[1];

2'b10: y = I[2];

2'b11: y = I[3];

endcase

endmodule

//stimulus

module test_mux ;

input  [3: 0] D;

input [1:0] S;

output Y;
```

```
//instantiate mux4_1_bh

mux4_1_bh uut ( .I(D), .S(select), .y(Y) );

//start simulation, provide stimulus to the module under test

initial

begin

D = 4'b0101;

S = 2'b00;

repeat (3)

initial
begin
  #10 S = S + 1; //increase S by 1

end

initial
begin

//monitor D, S, Y

 $monitor("%b \t %b \t %b, D, S, Y);

end

endmodule
```

5. Textbook 7.11: Obtain the 15-bit Hamming code word for the 11-bit data word 11001001010.

Position:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Notation:	P_1	P_2	B_3	P_4	B_5	B_6	B_7	P_8	B_9	B_{10}	B_{11}	B_{12}	B_{13}	B_{14}	B_{15}
Data:	_	_	1	_	1	0	0	_	1	0	0	1	0	1	0

Parity Data:

$$P_1 = XOR\ (3, 5, 7, 9, 11, 13, 15) = 101001 = 1$$
$$P_2 = XOR\ (3, 6, 7, 10, 11, 14, 15) = 1000010 = 0$$
$$P_4 = XOR\ (5, 6, 7, 12, 13, 14, 15) = 100110 = 1$$
$$P_8 = XOR\ (9, 10, 11, 12, 13, 14, 15) = 1001010 = 1$$

Hamming Code: 101110011001010

6. Textbook 7.10: Given the 8-bit data word 01011011, generate the 13-bit composite word for the Hamming code that corrects single errors and detects double errors.

Position:	1	2	3	4	5	6	7	8	9	10	11	12	13
Notation:	P_1	P_2	B_3	P_4	B_5	B_6	B_7	P_8	B_9	B_{10}	B_{11}	B_{12}	P_E
Data:	_	_	0	_	1	0	1	_	1	0	1	1	_

Parity:

$$P_1 = XOR\ (3, 5, 7, 9, 11) = 01111 = 0$$
$$P_2 = XOR\ (3, 6, 7, 10, 11) = 00101 = 0$$
$$P_4 = XOR\ (5, 6, 7, 12) = 1011 = 1$$
$$P_8 = XOR\ (9, 10, 11, 12) = 1011 = 1$$
$$P_E = Even\ or\ Odd\ parity\ on\ bit\ 1 - 12 = 000110111011? = 1$$

Hamming Code: 0001101110111