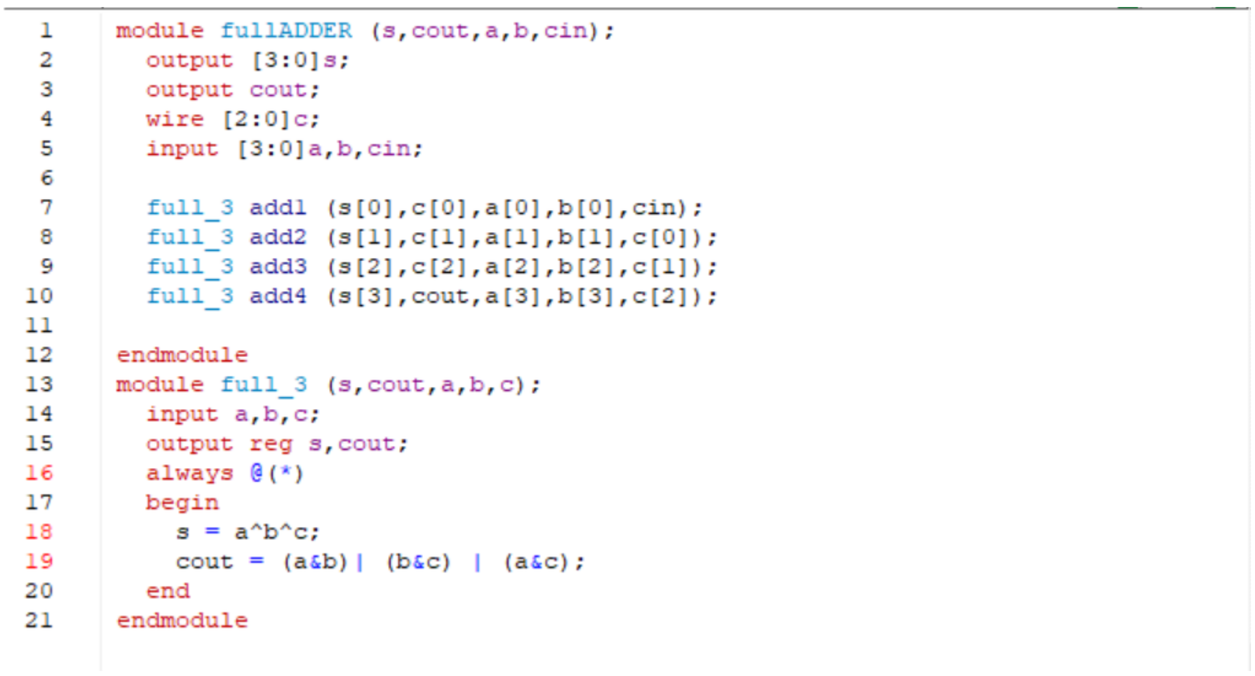
V. Evaluate my Learnings

Name: REIMARC G. CORPUZ Score: ____________

Course/Sec: BSCPE 3GF Time/Day: 5:46 TUESDAY

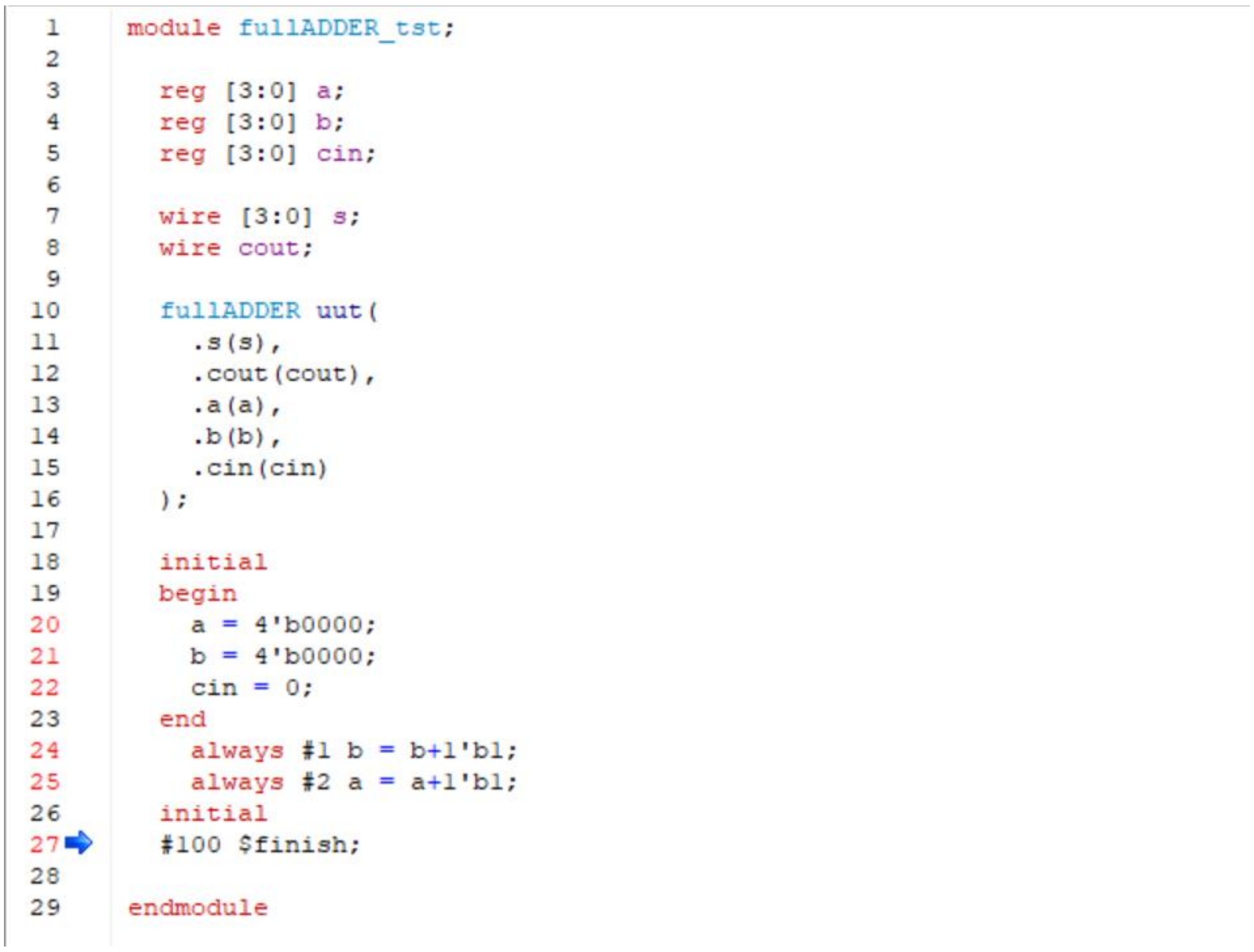
Date Sub: NOV. 15, 2022

A. write the Verilog code of the 4-bit full adder schematic shown in dataflow modeling and instantiation.



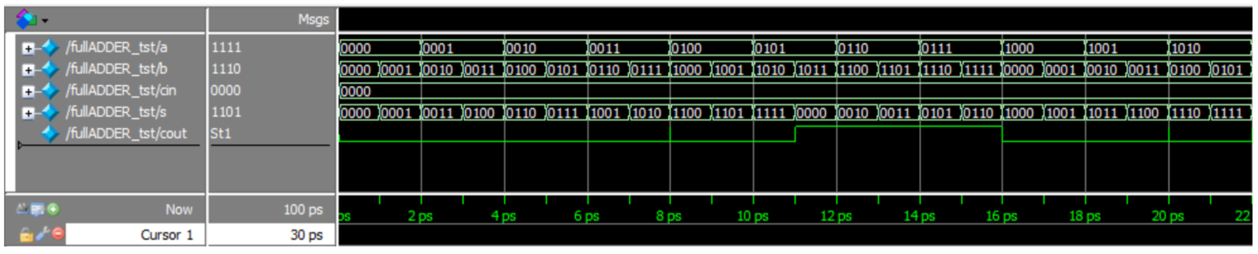
```
module fullADDER (s,cout,a,b,cin);
  output [3:0]s;
  output cout;
  wire [2:0]c;
  input [3:0]a,b,cin;

  full_3 add1 (s[0],c[0],a[0],b[0],cin);
  full_3 add2 (s[1],c[1],a[1],b[1],c[0]);
  full_3 add3 (s[2],c[2],a[2],b[2],c[1]);
  full_3 add4 (s[3],cout,a[3],b[3],c[2]);

endmodule
module full_3 (s,cout,a,b,c);
  input a,b,c;
  output reg s,cout;
  always @(*)
  begin
    s = a^b^c;
    cout = (a&b)| (b&c) | (a&c);
  end
endmodule
```

B. Modify and write the previous test bench in module 4, use this to test the 4-bit full adder.

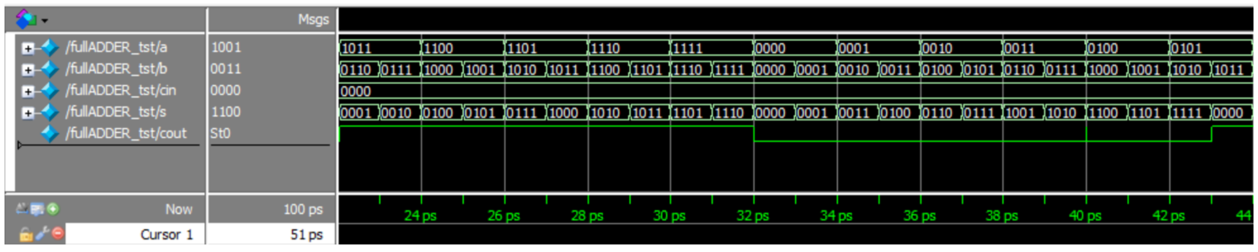


```
module fullADDER_tst;

  reg [3:0] a;
  reg [3:0] b;
  reg [3:0] cin;

  wire [3:0] s;
  wire cout;

  fullADDER uut(
    .s(s),
    .cout(cout),
    .a(a),
    .b(b),
    .cin(cin)
  );

  initial
  begin
    a = 4'b0000;
    b = 4'b0000;
    cin = 0;
  end
    always #1 b = b+1'b1;
    always #2 a = a+1'b1;
  initial
  #100 $finish;

endmodule
```

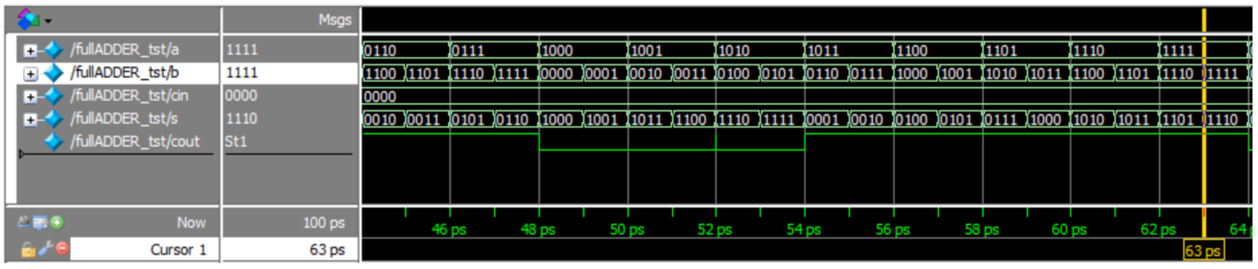
C. Draw the expected waveform here:



```
/fullADDER_tst/a      1111
/fullADDER_tst/b      1110
/fullADDER_tst/cin    0000
/fullADDER_tst/s      1101
/fullADDER_tst/cout   St1
Now                   100 ps
Cursor 1              30 ps
```



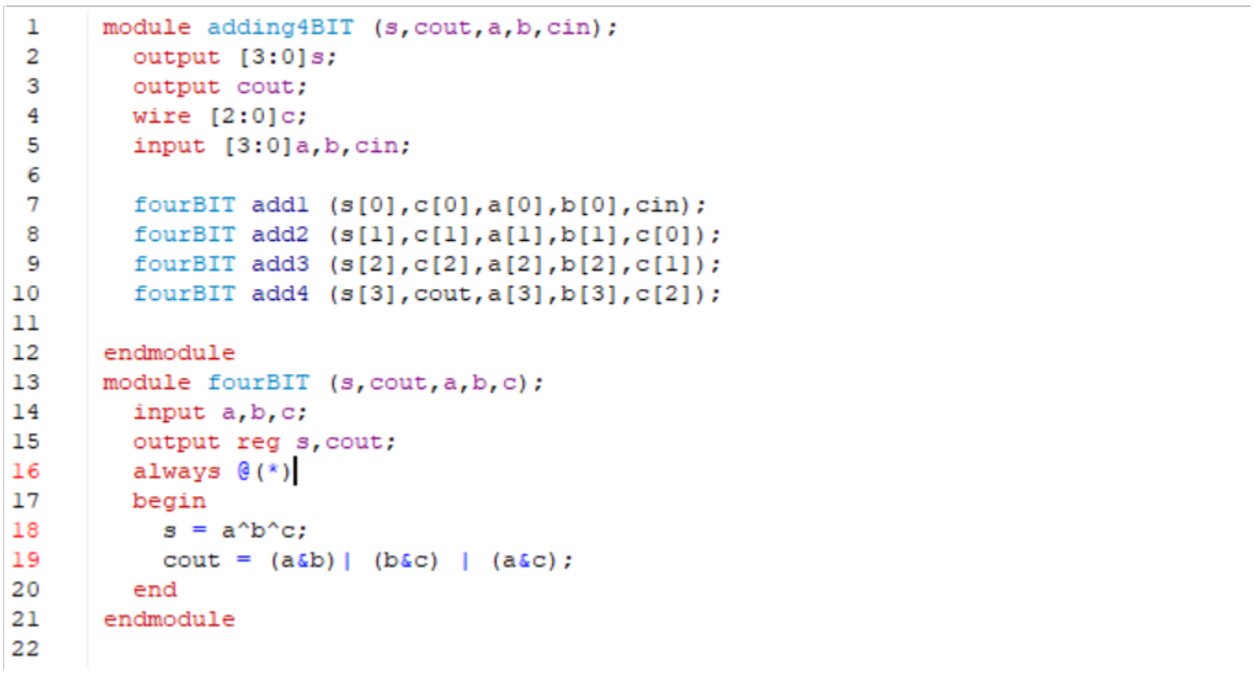
```
/fullADDER_tst/a      1001
/fullADDER_tst/b      0011
/fullADDER_tst/cin    0000
/fullADDER_tst/s      1100
/fullADDER_tst/cout   St0
Now                   100 ps
Cursor 1              51 ps
```



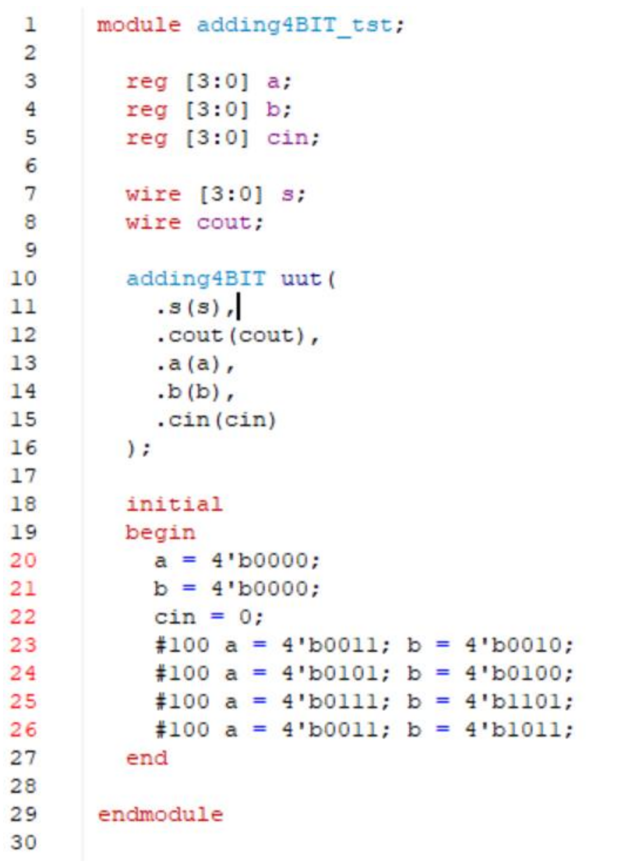
```
/fullADDER_tst/a      1111
/fullADDER_tst/b      1111
/fullADDER_tst/cin    0000
/fullADDER_tst/s      1110
/fullADDER_tst/cout   St1
Now                   100 ps
Cursor 1              63 ps
```

D. Test the 4bit adder circuit by adding the following 4-bit binary values. (Negative numbers are given in two's complement notation. Assume that the first expression is [3:0]A+[3:0]B= [3:0]SUM

		binary	decimal
3+2=5	0011+0010=	________	________
5+4=9	0101+0100=	________	________
7−3=4	0111+1101=	________	________
3−5=−2	0011+1011=	________	________

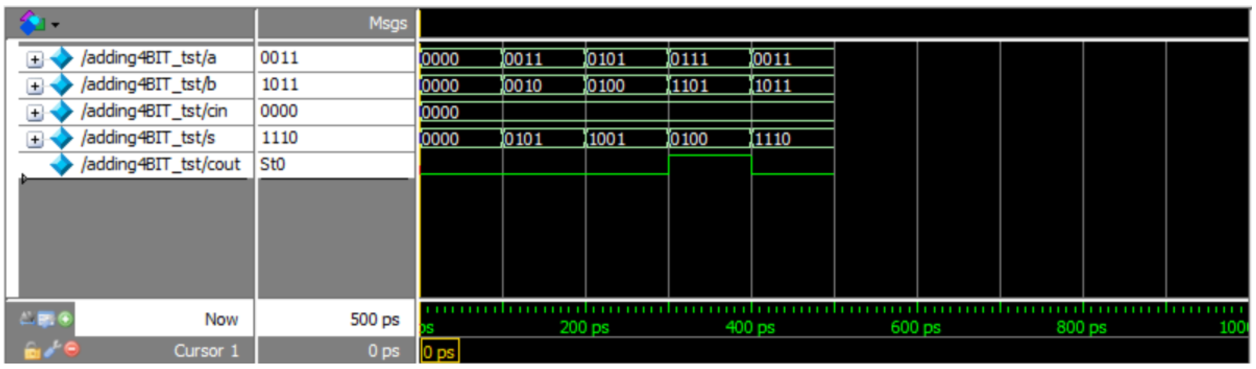


```
module adding4BIT (s,cout,a,b,cin);
  output [3:0]s;
  output cout;
  wire [2:0]c;
  input [3:0]a,b,cin;

  fourBIT add1 (s[0],c[0],a[0],b[0],cin);
  fourBIT add2 (s[1],c[1],a[1],b[1],c[0]);
  fourBIT add3 (s[2],c[2],a[2],b[2],c[1]);
  fourBIT add4 (s[3],cout,a[3],b[3],c[2]);

endmodule
module fourBIT (s,cout,a,b,c);
  input a,b,c;
  output reg s,cout;
  always @(*)
  begin
    s = a^b^c;
    cout = (a&b)| (b&c) | (a&c);
  end
endmodule
```



```
module adding4BIT_tst;

  reg [3:0] a;
  reg [3:0] b;
  reg [3:0] cin;

  wire [3:0] s;
  wire cout;

  adding4BIT uut(
    .s(s),
    .cout(cout),
    .a(a),
    .b(b),
    .cin(cin)
  );

  initial
  begin
    a = 4'b0000;
    b = 4'b0000;
    cin = 0;
    #100 a = 4'b0011; b = 4'b0010;
    #100 a = 4'b0101; b = 4'b0100;
    #100 a = 4'b0111; b = 4'b1101;
    #100 a = 4'b0011; b = 4'b1011;
  end

endmodule
```

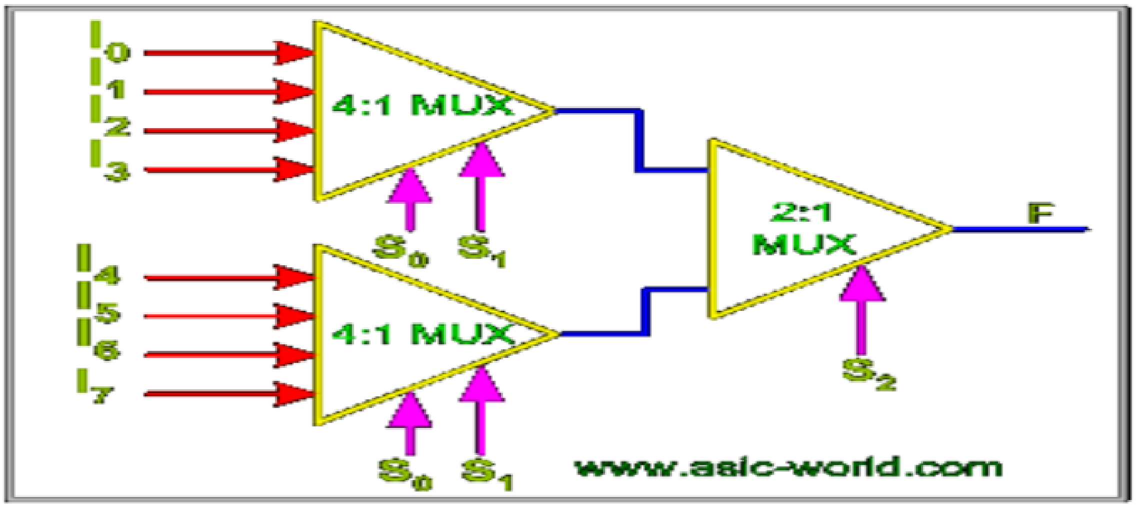
Draw the expected waveform



	Msgs
/adding4BIT_tst/a	0011
/adding4BIT_tst/b	1011
/adding4BIT_tst/cin	0000
/adding4BIT_tst/s	1110
/adding4BIT_tst/cout	St0
Now	500 ps
Cursor 1	0 ps

PART 2

Create a Verilog code for Dataflow and Structural for the circuit of 8:1 multiplexer. Illustrate the truth table of the following. Present the simulation result in a wave form corresponding to a test bench created.

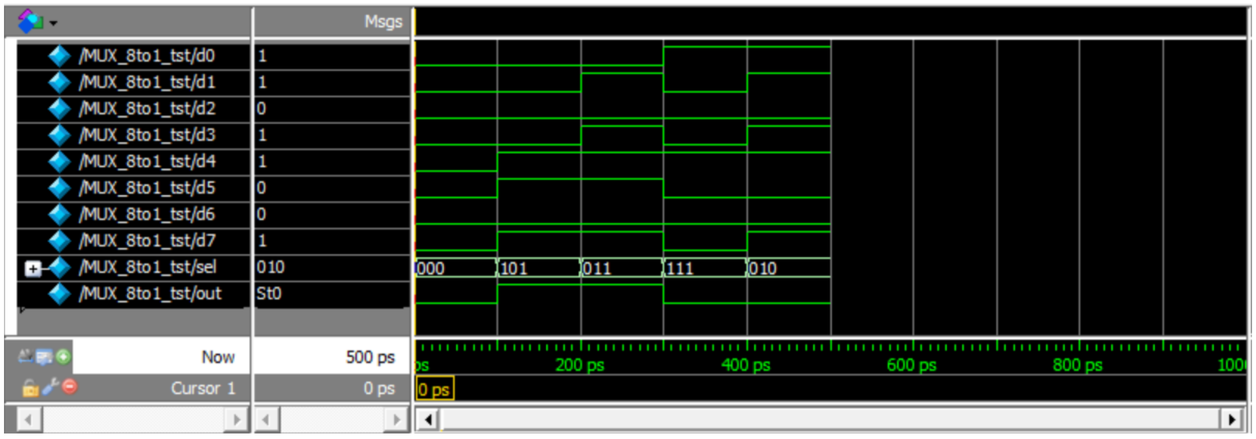


A. Show your truth table and waveform here:

Truth Table

S0	S1	S2	OUTPUT
0	0	0	I0
0	0	1	I1
0	1	0	I2
0	1	1	I3
1	0	0	I4
1	0	1	I5
1	1	0	I7
1	1	1	I7

Waveform



```
                      Msgs
/MUX_8to1_tst/d0      1
/MUX_8to1_tst/d1      1
/MUX_8to1_tst/d2      0
/MUX_8to1_tst/d3      1
/MUX_8to1_tst/d4      1
/MUX_8to1_tst/d5      0
/MUX_8to1_tst/d6      0
/MUX_8to1_tst/d7      1
/MUX_8to1_tst/sel     010      000  101  011  111  010
/MUX_8to1_tst/out     St0

Now                   500 ps
Cursor 1              0 ps
```

B. Create structural modeling of 8:1 multiplexer

```
module and_gate(output a, input b, c, d, e);
        assign a = b & c & d & e;
```

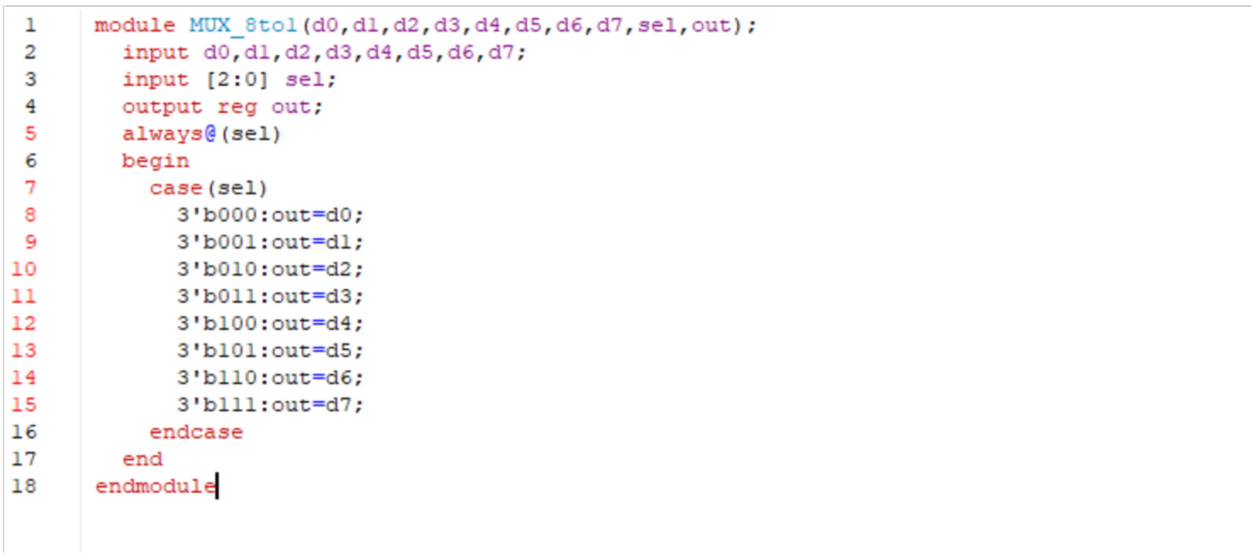
```
endmodule

module not_gate(output f, input g);
            assign f = ~ g;
endmodule

module or_gate(output l, input m, n, o, p, q, r, s, t);
            assign l = m | n | o | p | q | r | s | t;
endmodule

module m81(out, D0, D1, D2, D3, D4, D5, D6, D7, S0, S1, S2);
            output out;
            input D0, D1, D2, D3, D4, D5, D6, D7, S0, S1, S2;
            wire s0bar, s1bar, T1, T2, T3, T4, T5, T6, T7, T8;

            not_gate u1(s1bar, S1);
            not_gate u2(s0bar, S0);
            not_gate u3(s2bar, S2);

            and_gate u4(T1, D0, s0bar, s1bar, s2bar);
            and_gate u5(T2, D1, S0, s1bar, s2bar);
            and_gate u6(T3, D2, s0bar, S1, s2bar);
            and_gate u7(T4, D3, S0, S1, s2bar);
            and_gate u8(T5, D4, s0bar, s1bar, S2);
            and_gate u9(T6, D5, S0, s1bar, S2);
            and_gate u10(T7, D6, s0bar, S1, S2);
            and_gate u11(T8, D7, S0, S1, S2);
            or_gate u12(out, T1, T2, T3, T4, T5, T6, T7, T8);
endmodule
```

C. Create dataflow modeling of 8:1 multiplexer

```
module m81(output out, input D0, D1, D2, D3, D4, D5, D6, D7, S0, S1, S2);
            assign S1bar=~S1;
            assign S0bar=~S0;
            assign S2bar=~S2;
            assign out = (D0 & S2bar & S1bar & S0bar) | (D1 & S2bar & S1bar & S0) | (D2 & S2bar &
            S1 & S0bar) + (D3 & S2bar & S1 & S0) + (D4 & S2 & S1bar & S0bar) + (D5 & S2 & S1bar &
            S0) + (D6 & S2 & S1 & S0bar) + (D7 & S2 & S1 & S0);
endmodule
```

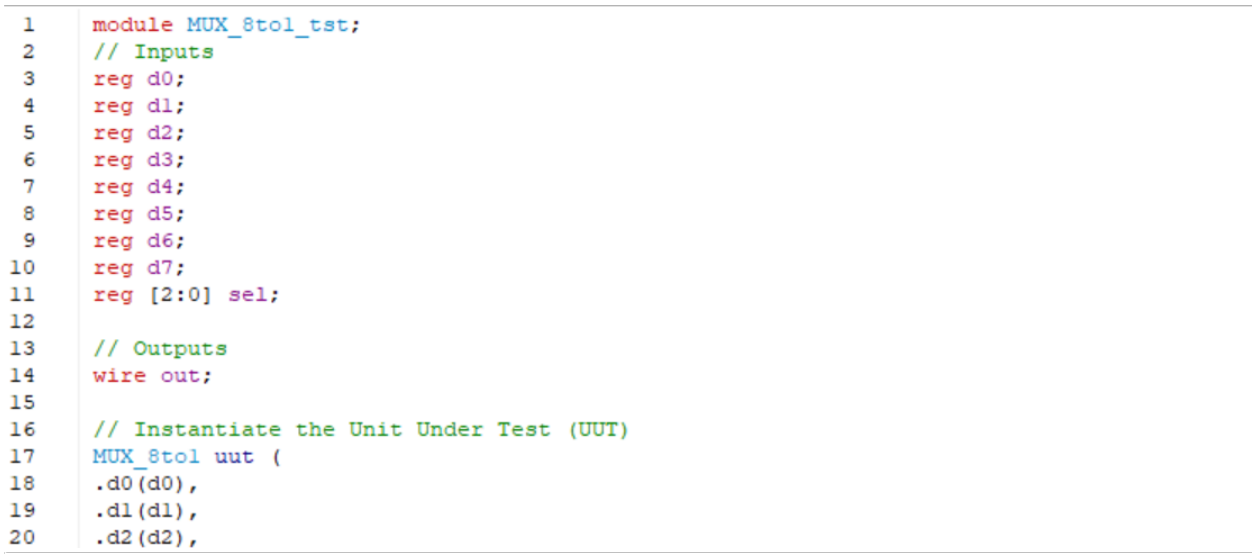
D. Create a Test Bench Simulation to the following 8:1 multiplexer

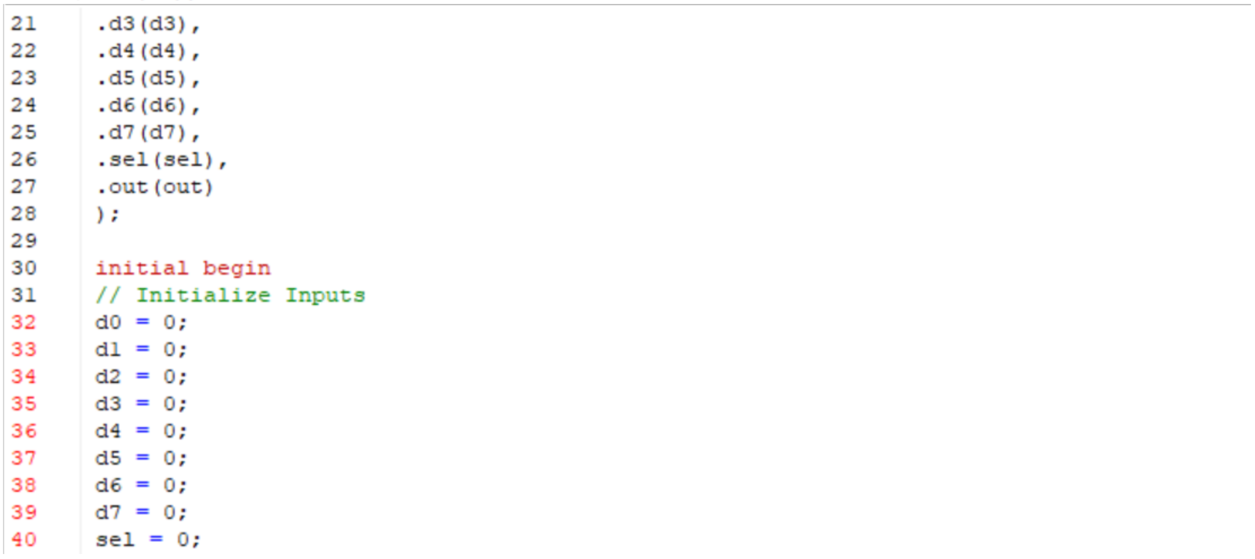
Verilog Code



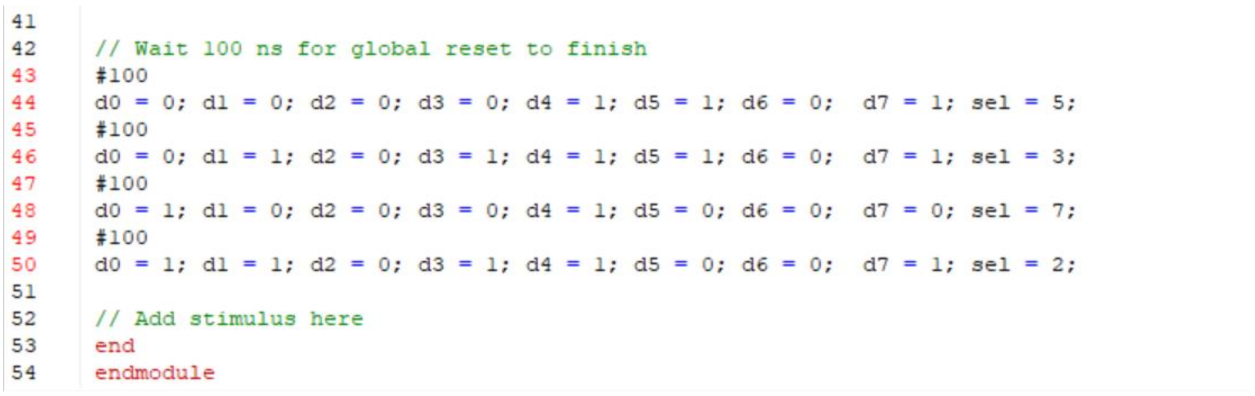
```verilog
module MUX_8to1(d0,d1,d2,d3,d4,d5,d6,d7,sel,out);
  input d0,d1,d2,d3,d4,d5,d6,d7;
  input [2:0] sel;
  output reg out;
  always@(sel)
  begin
    case(sel)
      3'b000:out=d0;
      3'b001:out=d1;
      3'b010:out=d2;
      3'b011:out=d3;
      3'b100:out=d4;
      3'b101:out=d5;
      3'b110:out=d6;
      3'b111:out=d7;
    endcase
  end
endmodule
```

Testbench





```verilog
module MUX_8to1_tst;
// Inputs
reg d0;
reg d1;
reg d2;
reg d3;
reg d4;
reg d5;
reg d6;
reg d7;
reg [2:0] sel;

// Outputs
wire out;

// Instantiate the Unit Under Test (UUT)
MUX_8to1 uut (
.d0(d0),
.d1(d1),
.d2(d2),
.d3(d3),
.d4(d4),
.d5(d5),
.d6(d6),
.d7(d7),
.sel(sel),
.out(out)
);

initial begin
// Initialize Inputs
d0 = 0;
d1 = 0;
d2 = 0;
d3 = 0;
d4 = 0;
d5 = 0;
d6 = 0;
d7 = 0;
sel = 0;
```



```

// Wait 100 ns for global reset to finish
#100
d0 = 0; d1 = 0; d2 = 0; d3 = 0; d4 = 1; d5 = 1; d6 = 0;  d7 = 1; sel = 5;
#100
d0 = 0; d1 = 1; d2 = 0; d3 = 1; d4 = 1; d5 = 1; d6 = 0;  d7 = 1; sel = 3;
#100
d0 = 1; d1 = 0; d2 = 0; d3 = 0; d4 = 1; d5 = 0; d6 = 0;  d7 = 0; sel = 7;
#100
d0 = 1; d1 = 1; d2 = 0; d3 = 1; d4 = 1; d5 = 0; d6 = 0;  d7 = 1; sel = 2;

// Add stimulus here
end
endmodule
```