CS-223

Digital Design

Section: 3

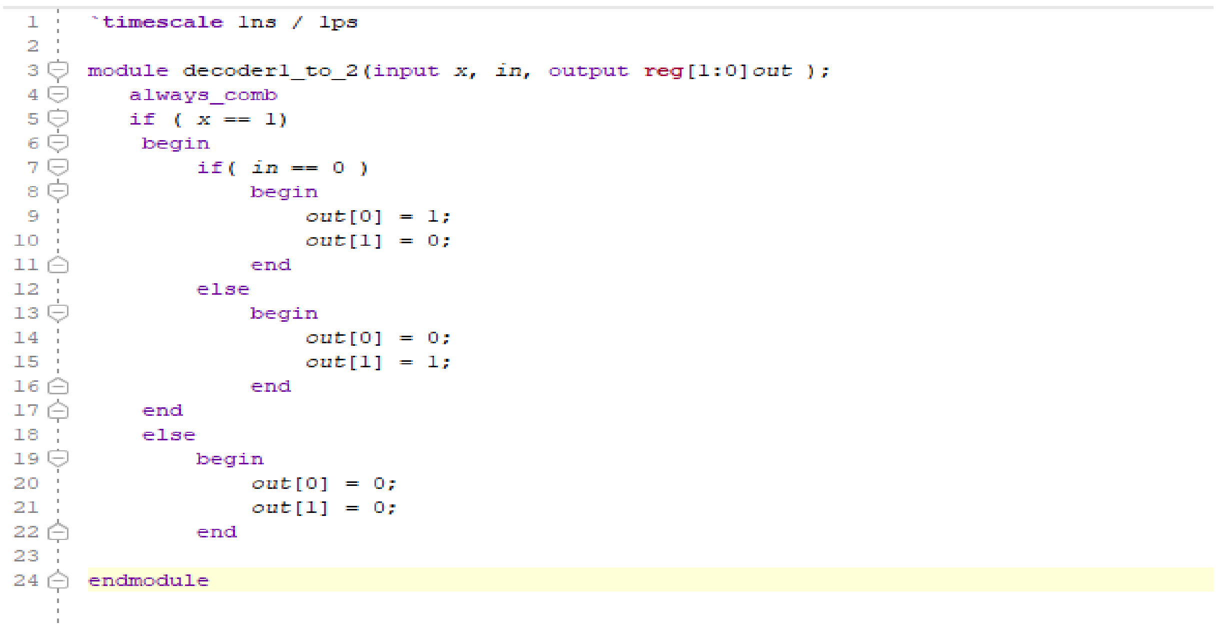
LAB 3

Mehmet Hasat Serinkan

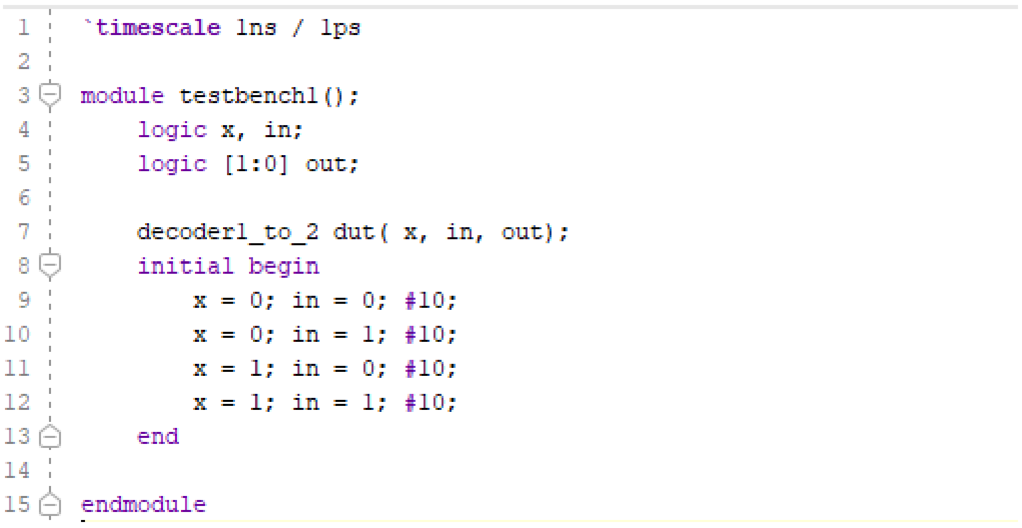
21901649

13.03.2022

B) Behavioral SystemVerilog module for a 1-to-2 decoder (including enable signal) and a testbench for it.

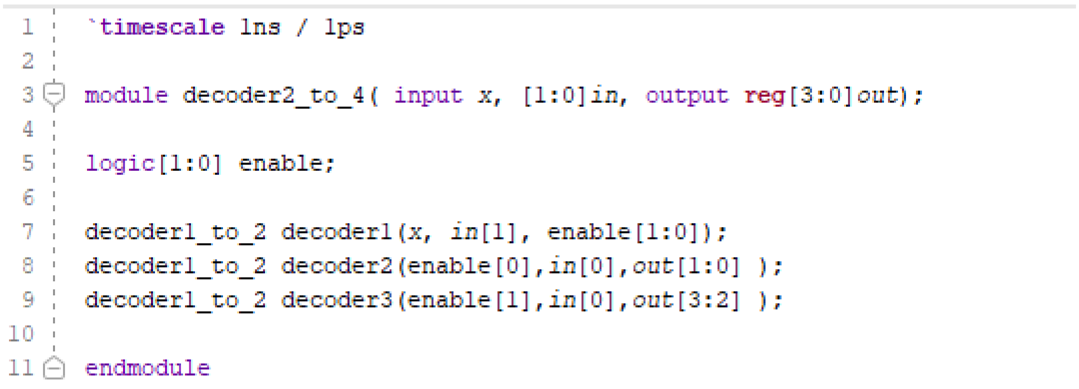


```
`timescale 1ns / 1ps

module decoder1_to_2(input x, in, output reg[1:0]out );
   always_comb
   if ( x == 1)
    begin
        if( in == 0 )
            begin
                out[0] = 1;
                out[1] = 0;
            end
        else
            begin
                out[0] = 0;
                out[1] = 1;
            end
    end
    else
        begin
            out[0] = 0;
            out[1] = 0;
        end

endmodule
```

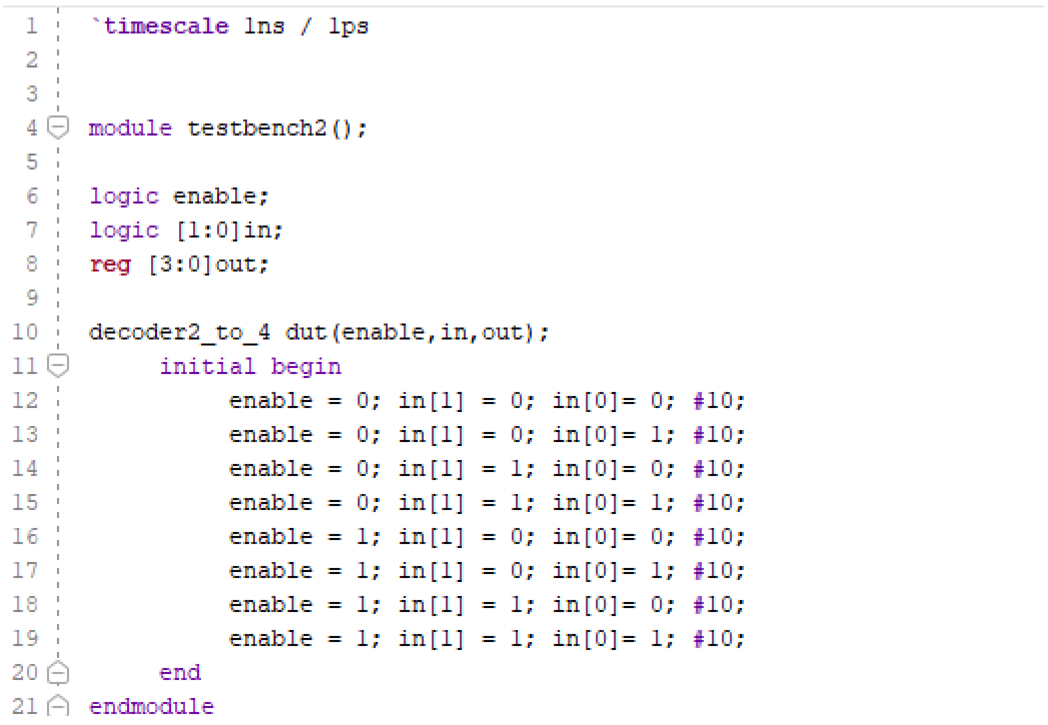


```
`timescale 1ns / 1ps

module testbench1();
    logic x, in;
    logic [1:0] out;

    decoder1_to_2 dut( x, in, out);
    initial begin
        x = 0; in = 0; #10;
        x = 0; in = 1; #10;
        x = 1; in = 0; #10;
        x = 1; in = 1; #10;
    end

endmodule
```

C) Structural SystemVerilog module for a 2-to-4 decoder (including enable signal) using three 1-to-2 decoders. Prepare a testbench for it.

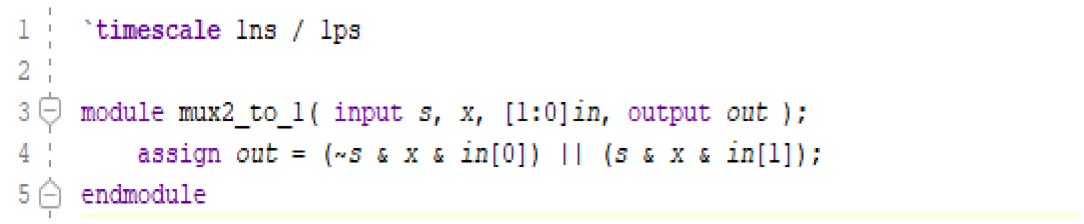


```
`timescale 1ns / 1ps

module decoder2_to_4( input x, [1:0]in, output reg[3:0]out);

logic[1:0] enable;

decoder1_to_2 decoder1(x, in[1], enable[1:0]);
decoder1_to_2 decoder2(enable[0],in[0],out[1:0] );
decoder1_to_2 decoder3(enable[1],in[0],out[3:2] );

endmodule
```



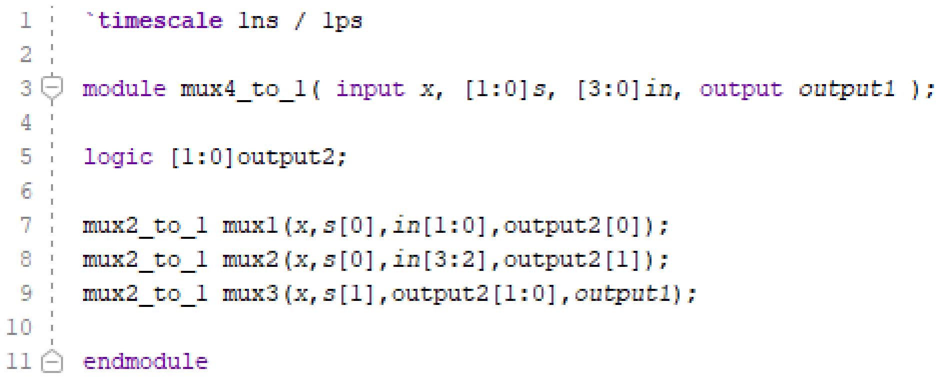
```
`timescale 1ns / 1ps


module testbench2();

logic enable;
logic [1:0]in;
reg [3:0]out;

decoder2_to_4 dut(enable,in,out);
     initial begin
          enable = 0; in[1] = 0; in[0]= 0; #10;
          enable = 0; in[1] = 0; in[0]= 1; #10;
          enable = 0; in[1] = 1; in[0]= 0; #10;
          enable = 0; in[1] = 1; in[0]= 1; #10;
          enable = 1; in[1] = 0; in[0]= 0; #10;
          enable = 1; in[1] = 0; in[0]= 1; #10;
          enable = 1; in[1] = 1; in[0]= 0; #10;
          enable = 1; in[1] = 1; in[0]= 1; #10;
     end
endmodule
```

D) Behavioral SystemVerilog module for a 2-to-1 multiplexer.

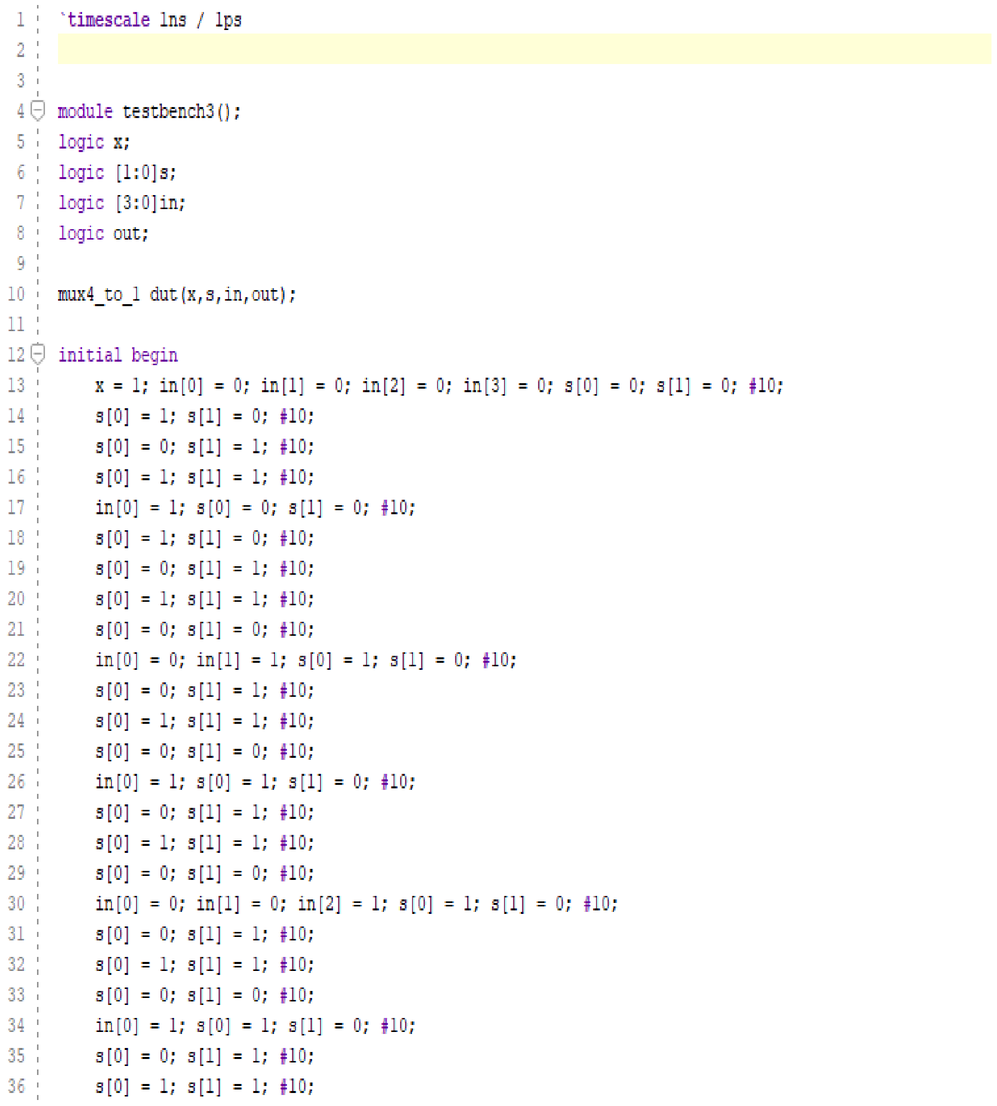


```
`timescale 1ns / 1ps

module mux2_to_1( input s, x, [1:0]in, output out );
    assign out = (~s & x & in[0]) || (s & x & in[1]);
endmodule
```

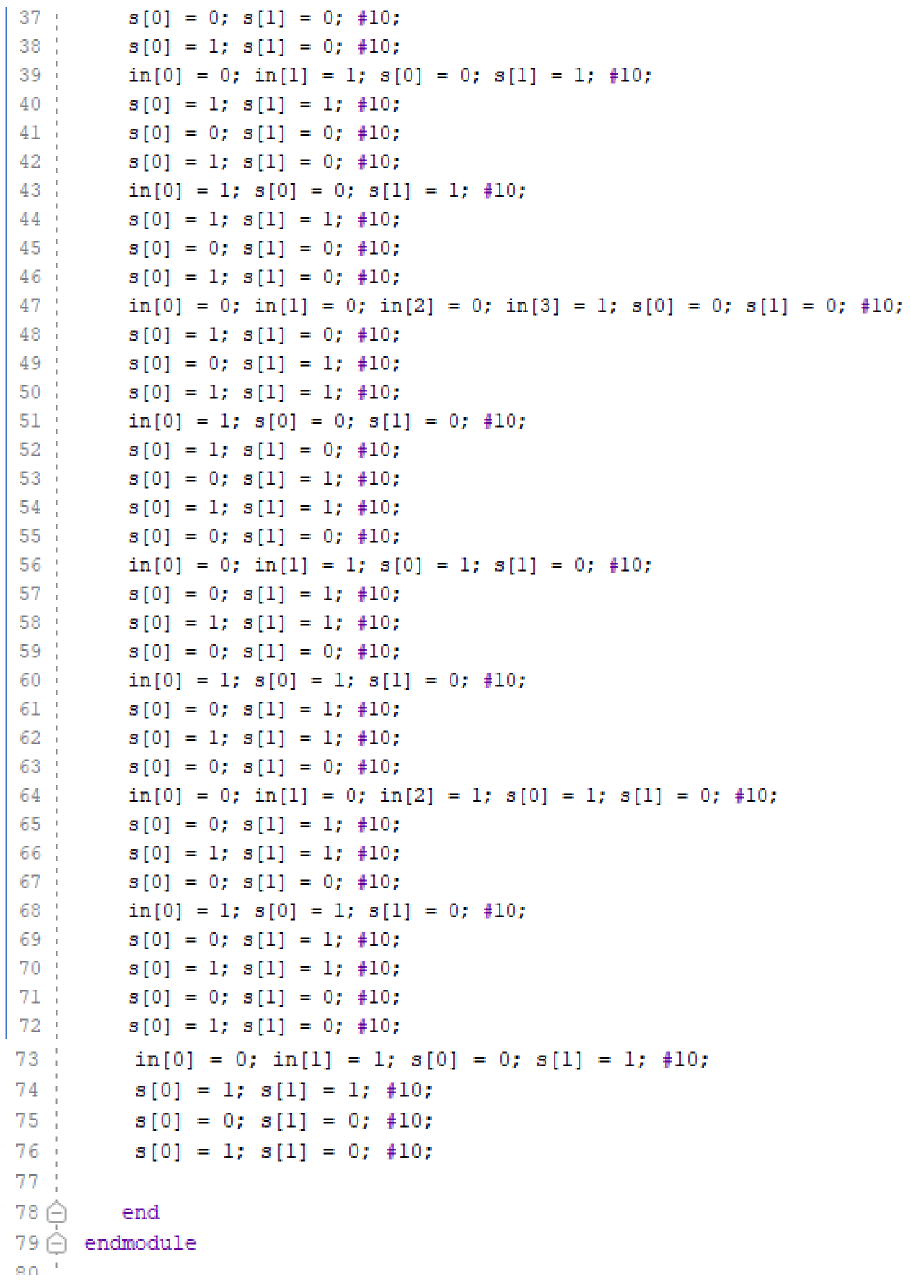
E) Structural SystemVerilog module for a 4-to-1 multiplexer using three 2-to-1 multiplexers. Prepare a testbench for it.



```
`timescale 1ns / 1ps

module mux4_to_1( input x, [1:0]s, [3:0]in, output output1 );

logic [1:0]output2;

mux2_to_1 mux1(x,s[0],in[1:0],output2[0]);
mux2_to_1 mux2(x,s[0],in[3:2],output2[1]);
mux2_to_1 mux3(x,s[1],output2[1:0],output1);

endmodule
```

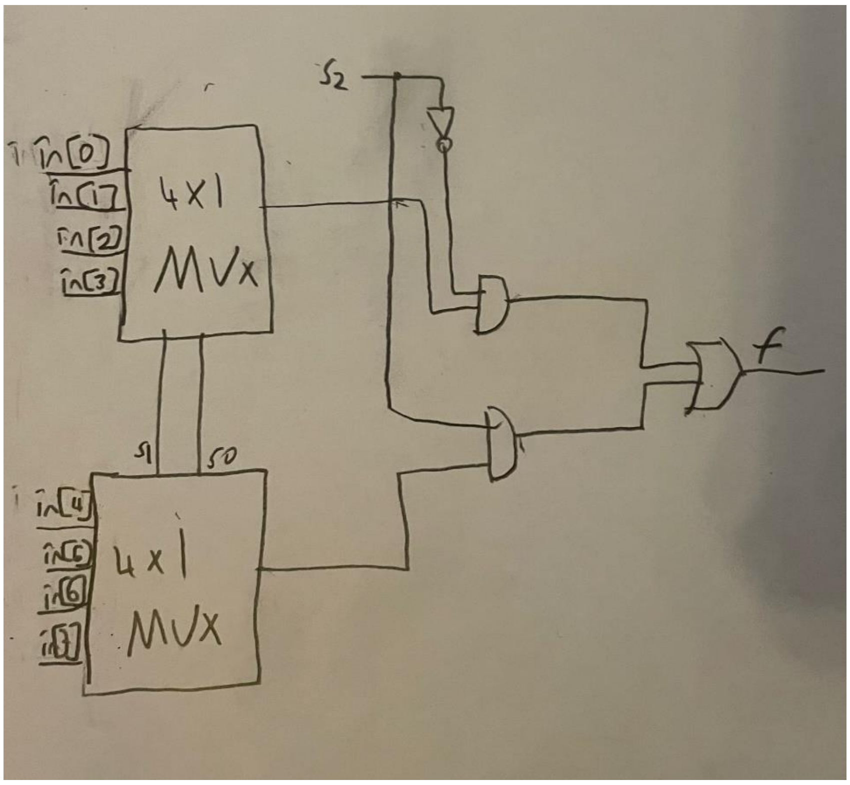


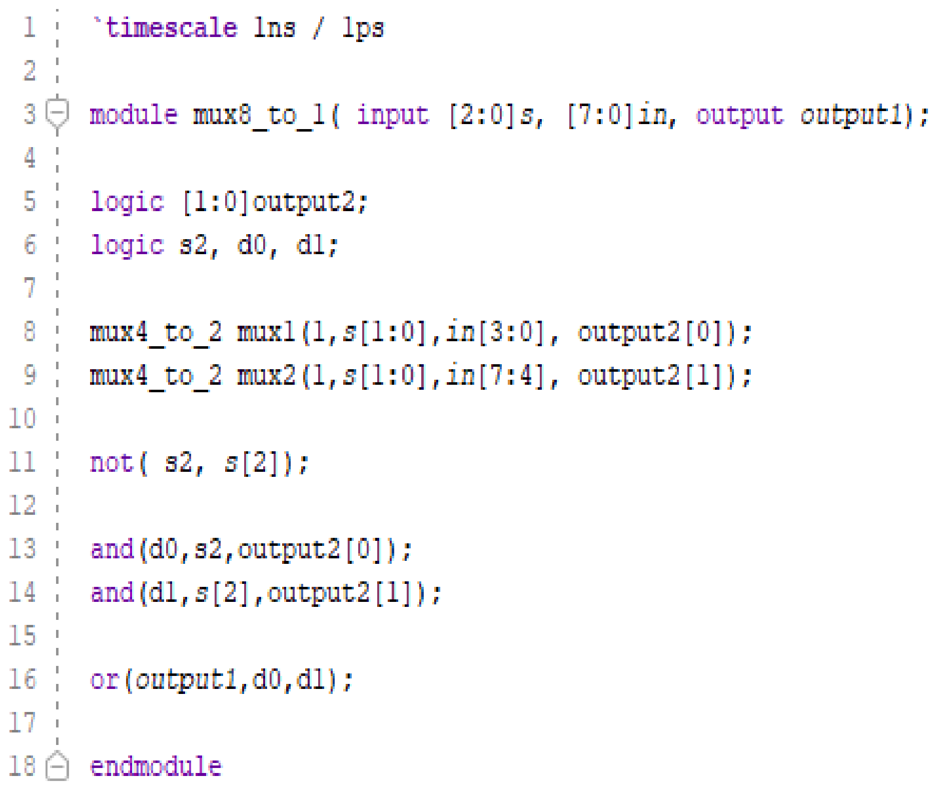
```
`timescale 1ns / 1ps


module testbench3();
logic x;
logic [1:0]s;
logic [3:0]in;
logic out;

mux4_to_1 dut(x,s,in,out);

initial begin
    x = 1; in[0] = 0; in[1] = 0; in[2] = 0; in[3] = 0; s[0] = 0; s[1] = 0; #10;
    s[0] = 1; s[1] = 0; #10;
    s[0] = 0; s[1] = 1; #10;
    s[0] = 1; s[1] = 1; #10;
    in[0] = 1; s[0] = 0; s[1] = 0; #10;
    s[0] = 1; s[1] = 0; #10;
    s[0] = 0; s[1] = 1; #10;
    s[0] = 1; s[1] = 1; #10;
    s[0] = 0; s[1] = 0; #10;
    in[0] = 0; in[1] = 1; s[0] = 1; s[1] = 0; #10;
    s[0] = 0; s[1] = 1; #10;
    s[0] = 1; s[1] = 1; #10;
    s[0] = 0; s[1] = 0; #10;
    in[0] = 1; s[0] = 1; s[1] = 0; #10;
    s[0] = 0; s[1] = 1; #10;
    s[0] = 1; s[1] = 1; #10;
    s[0] = 0; s[1] = 0; #10;
    in[0] = 0; in[1] = 0; in[2] = 1; s[0] = 1; s[1] = 0; #10;
    s[0] = 0; s[1] = 1; #10;
    s[0] = 1; s[1] = 1; #10;
    s[0] = 0; s[1] = 0; #10;
    in[0] = 1; s[0] = 1; s[1] = 0; #10;
    s[0] = 0; s[1] = 1; #10;
    s[0] = 1; s[1] = 1; #10;
```



```verilog
    s[0] = 0; s[1] = 0; #10;
    s[0] = 1; s[1] = 0; #10;
    in[0] = 0; in[1] = 1; s[0] = 0; s[1] = 1; #10;
    s[0] = 1; s[1] = 1; #10;
    s[0] = 0; s[1] = 0; #10;
    s[0] = 1; s[1] = 0; #10;
    in[0] = 1; s[0] = 0; s[1] = 1; #10;
    s[0] = 1; s[1] = 1; #10;
    s[0] = 0; s[1] = 0; #10;
    s[0] = 1; s[1] = 0; #10;
    in[0] = 0; in[1] = 0; in[2] = 0; in[3] = 1; s[0] = 0; s[1] = 0; #10;
    s[0] = 1; s[1] = 0; #10;
    s[0] = 0; s[1] = 1; #10;
    s[0] = 1; s[1] = 1; #10;
    in[0] = 1; s[0] = 0; s[1] = 0; #10;
    s[0] = 1; s[1] = 0; #10;
    s[0] = 0; s[1] = 1; #10;
    s[0] = 1; s[1] = 1; #10;
    s[0] = 0; s[1] = 0; #10;
    in[0] = 0; in[1] = 1; s[0] = 1; s[1] = 0; #10;
    s[0] = 0; s[1] = 1; #10;
    s[0] = 1; s[1] = 1; #10;
    s[0] = 0; s[1] = 0; #10;
    in[0] = 1; s[0] = 1; s[1] = 0; #10;
    s[0] = 0; s[1] = 1; #10;
    s[0] = 1; s[1] = 1; #10;
    s[0] = 0; s[1] = 0; #10;
    in[0] = 0; in[1] = 0; in[2] = 1; s[0] = 1; s[1] = 0; #10;
    s[0] = 0; s[1] = 1; #10;
    s[0] = 1; s[1] = 1; #10;
    s[0] = 0; s[1] = 0; #10;
    in[0] = 1; s[0] = 1; s[1] = 0; #10;
    s[0] = 0; s[1] = 1; #10;
    s[0] = 1; s[1] = 1; #10;
    s[0] = 0; s[1] = 0; #10;
    s[0] = 1; s[1] = 0; #10;
    in[0] = 0; in[1] = 1; s[0] = 0; s[1] = 1; #10;
    s[0] = 1; s[1] = 1; #10;
    s[0] = 0; s[1] = 0; #10;
    s[0] = 1; s[1] = 0; #10;

   end
endmodule
```

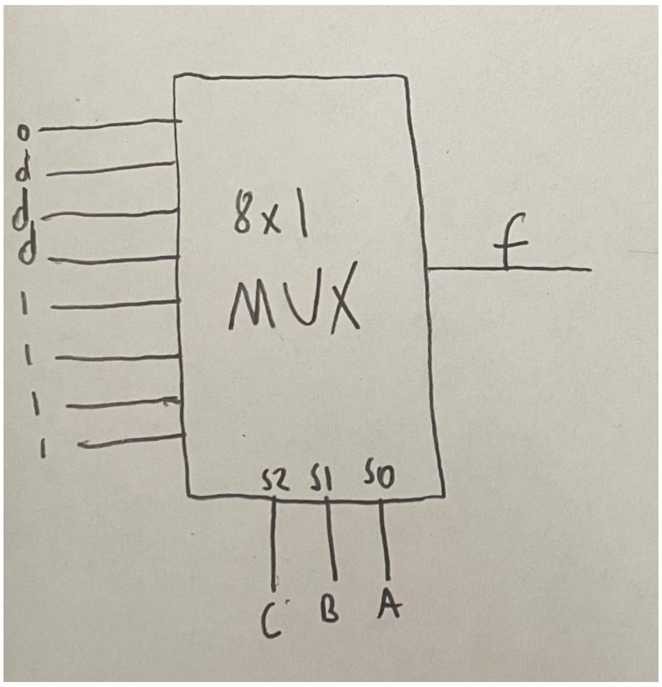
F) Block diagram and structural System Verilog module of 8-to-1 MUX by using two 4-to-1 MUX modules, two AND gates, an INVERTER, and an OR gate.

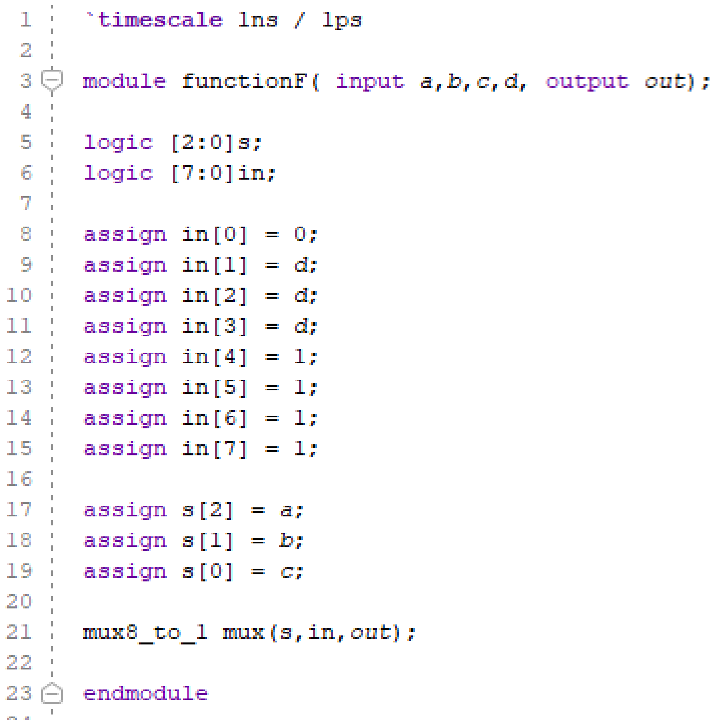




```
`timescale 1ns / 1ps

module mux8_to_1( input [2:0]s, [7:0]in, output output1);

logic [1:0]output2;
logic s2, d0, d1;

mux4_to_2 mux1(1,s[1:0],in[3:0], output2[0]);
mux4_to_2 mux2(1,s[1:0],in[7:4], output2[1]);

not( s2, s[2]);

and(d0,s2,output2[0]);
and(d1,s[2],output2[1]);

or(output1,d0,d1);

endmodule
```

G) Block diagram and SystemVerilog module for the function F, using only one 8-to-1 multiplexer and an INVERTER (if you wish). The function F is as follows:
F(A,B,C,D)= {1 if 10<$ABCD$+$DABC$<200 $otherwise$
where ABCD and DABC are 4-bit unsigned binary numbers. For ABCD, A is the most significant bit and D is the least significant bit. For DABC, D is the most significant bit and C is the least significant bit. E.g. F(0,1,0,1)=1 since ABCD=0101, DABC=1010, and ABCD+DABC=15 that results in 10 < ABCD+DABC < 20.




```
`timescale 1ns / 1ps

module functionF( input a,b,c,d, output out);

logic [2:0]s;
logic [7:0]in;

assign in[0] = 0;
assign in[1] = d;
assign in[2] = d;
assign in[3] = d;
assign in[4] = 1;
assign in[5] = 1;
assign in[6] = 1;
assign in[7] = 1;

assign s[2] = a;
assign s[1] = b;
assign s[0] = c;

mux8_to_1 mux(s,in,out);

endmodule
```