

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.

```
1  `timescale 1ns / 1ps
2
3  module decoder1_to_2(input x, in, output reg[1:0]out );
4      always_comb
5      if ( x == 1)
6      begin
7          if( in == 0 )
8          begin
9              out[0] = 1;
10             out[1] = 0;
11         end
12         else
13         begin
14             out[0] = 0;
15             out[1] = 1;
16         end
17     end
18     else
19     begin
20         out[0] = 0;
21         out[1] = 0;
22     end
23
24 endmodule
```

```
1  `timescale 1ns / 1ps
2
3  module testbench1();
4      logic x, in;
5      logic [1:0] out;
6
7      decoder1_to_2 dut( x, in, out);
8      initial begin
9          x = 0; in = 0; #10;
10         x = 0; in = 1; #10;
11         x = 1; in = 0; #10;
12         x = 1; in = 1; #10;
13     end
14
15 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.

```
1 | `timescale 1ns / 1ps
2 |
3 | module decoder2_to_4( input x, [1:0]in, output reg[3:0]out);
4 |
5 |     logic[1:0] enable;
6 |
7 |     decoder1_to_2 decoder1(x, in[1], enable[1:0]);
8 |     decoder1_to_2 decoder2(enable[0],in[0],out[1:0] );
9 |     decoder1_to_2 decoder3(enable[1],in[0],out[3:2] );
10 |
11 | endmodule
```

```
1 | `timescale 1ns / 1ps
2 |
3 |
4 | module testbench2();
5 |
6 |     logic enable;
7 |     logic [1:0]in;
8 |     reg [3:0]out;
9 |
10 |    decoder2_to_4 dut(enable,in,out);
11 |    initial begin
12 |        enable = 0; in[1] = 0; in[0]= 0; #10;
13 |        enable = 0; in[1] = 0; in[0]= 1; #10;
14 |        enable = 0; in[1] = 1; in[0]= 0; #10;
15 |        enable = 0; in[1] = 1; in[0]= 1; #10;
16 |        enable = 1; in[1] = 0; in[0]= 0; #10;
17 |        enable = 1; in[1] = 0; in[0]= 1; #10;
18 |        enable = 1; in[1] = 1; in[0]= 0; #10;
19 |        enable = 1; in[1] = 1; in[0]= 1; #10;
20 |    end
21 | endmodule
```

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

```
1 | `timescale 1ns / 1ps
2 |
3 | module mux2_to_1( input s, x, [1:0]in, output out );
4 |     assign out = (~s & x & in[0]) || (s & x & in[1]);
5 | endmodule
```

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

```

1  `timescale 1ns / 1ps
2
3  module mux4_to_1( input x, [1:0]s, [3:0]in, output output1 );
4
5      logic [1:0]output2;
6
7      mux2_to_1 mux1(x,s[0],in[1:0],output2[0]);
8      mux2_to_1 mux2(x,s[0],in[3:2],output2[1]);
9      mux2_to_1 mux3(x,s[1],output2[1:0],output1);
10
11 endmodule

```

```

1  `timescale 1ns / 1ps
2
3
4  module testbench3();
5      logic x;
6      logic [1:0]s;
7      logic [3:0]in;
8      logic out;
9
10     mux4_to_1 dut(x,s,in,out);
11
12     initial begin
13         x = 1; in[0] = 0; in[1] = 0; in[2] = 0; in[3] = 0; s[0] = 0; s[1] = 0; #10;
14         s[0] = 1; s[1] = 0; #10;
15         s[0] = 0; s[1] = 1; #10;
16         s[0] = 1; s[1] = 1; #10;
17         in[0] = 1; s[0] = 0; s[1] = 0; #10;
18         s[0] = 1; s[1] = 0; #10;
19         s[0] = 0; s[1] = 1; #10;
20         s[0] = 1; s[1] = 1; #10;
21         s[0] = 0; s[1] = 0; #10;
22         in[0] = 0; in[1] = 1; s[0] = 1; s[1] = 0; #10;
23         s[0] = 0; s[1] = 1; #10;
24         s[0] = 1; s[1] = 1; #10;
25         s[0] = 0; s[1] = 0; #10;
26         in[0] = 1; s[0] = 1; s[1] = 0; #10;
27         s[0] = 0; s[1] = 1; #10;
28         s[0] = 1; s[1] = 1; #10;
29         s[0] = 0; s[1] = 0; #10;
30         in[0] = 0; in[1] = 0; in[2] = 1; s[0] = 1; s[1] = 0; #10;
31         s[0] = 0; s[1] = 1; #10;
32         s[0] = 1; s[1] = 1; #10;
33         s[0] = 0; s[1] = 0; #10;
34         in[0] = 1; s[0] = 1; s[1] = 0; #10;
35         s[0] = 0; s[1] = 1; #10;
36         s[0] = 1; s[1] = 1; #10;

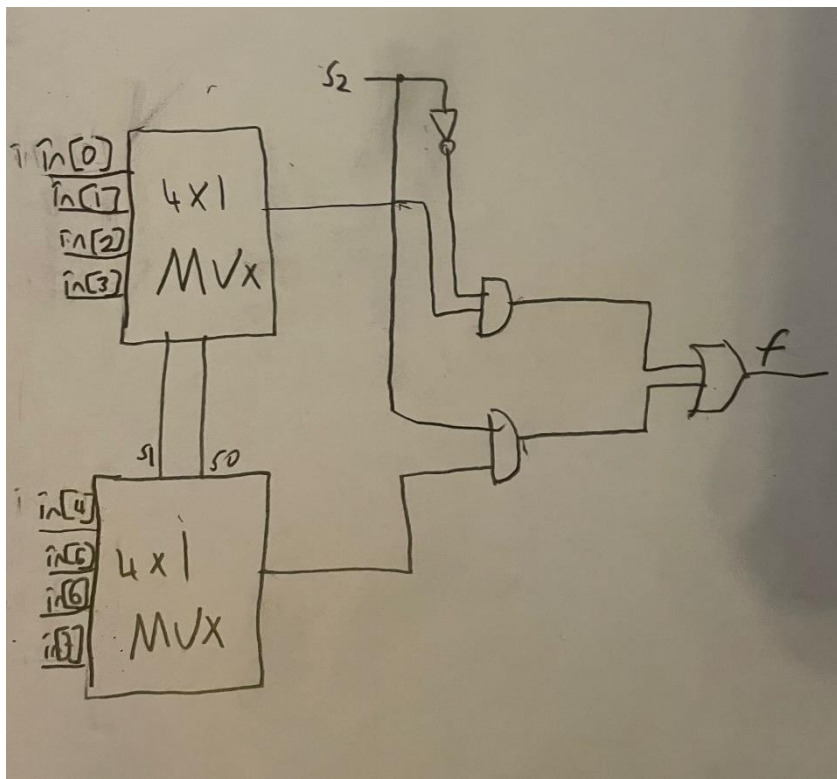
```

```

37     s[0] = 0; s[1] = 0; #10;
38     s[0] = 1; s[1] = 0; #10;
39     in[0] = 0; in[1] = 1; s[0] = 0; s[1] = 1; #10;
40     s[0] = 1; s[1] = 1; #10;
41     s[0] = 0; s[1] = 0; #10;
42     s[0] = 1; s[1] = 0; #10;
43     in[0] = 1; s[0] = 0; s[1] = 1; #10;
44     s[0] = 1; s[1] = 1; #10;
45     s[0] = 0; s[1] = 0; #10;
46     s[0] = 1; s[1] = 0; #10;
47     in[0] = 0; in[1] = 0; in[2] = 0; in[3] = 1; s[0] = 0; s[1] = 0; #10;
48     s[0] = 1; s[1] = 0; #10;
49     s[0] = 0; s[1] = 1; #10;
50     s[0] = 1; s[1] = 1; #10;
51     in[0] = 1; s[0] = 0; s[1] = 0; #10;
52     s[0] = 1; s[1] = 0; #10;
53     s[0] = 0; s[1] = 1; #10;
54     s[0] = 1; s[1] = 1; #10;
55     s[0] = 0; s[1] = 0; #10;
56     in[0] = 0; in[1] = 1; s[0] = 1; s[1] = 0; #10;
57     s[0] = 0; s[1] = 1; #10;
58     s[0] = 1; s[1] = 1; #10;
59     s[0] = 0; s[1] = 0; #10;
60     in[0] = 1; s[0] = 1; s[1] = 0; #10;
61     s[0] = 0; s[1] = 1; #10;
62     s[0] = 1; s[1] = 1; #10;
63     s[0] = 0; s[1] = 0; #10;
64     in[0] = 0; in[1] = 0; in[2] = 1; s[0] = 1; s[1] = 0; #10;
65     s[0] = 0; s[1] = 1; #10;
66     s[0] = 1; s[1] = 1; #10;
67     s[0] = 0; s[1] = 0; #10;
68     in[0] = 1; s[0] = 1; s[1] = 0; #10;
69     s[0] = 0; s[1] = 1; #10;
70     s[0] = 1; s[1] = 1; #10;
71     s[0] = 0; s[1] = 0; #10;
72     s[0] = 1; s[1] = 0; #10;
73     in[0] = 0; in[1] = 1; s[0] = 0; s[1] = 1; #10;
74     s[0] = 1; s[1] = 1; #10;
75     s[0] = 0; s[1] = 0; #10;
76     s[0] = 1; s[1] = 0; #10;
77
78     end
79 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.



```

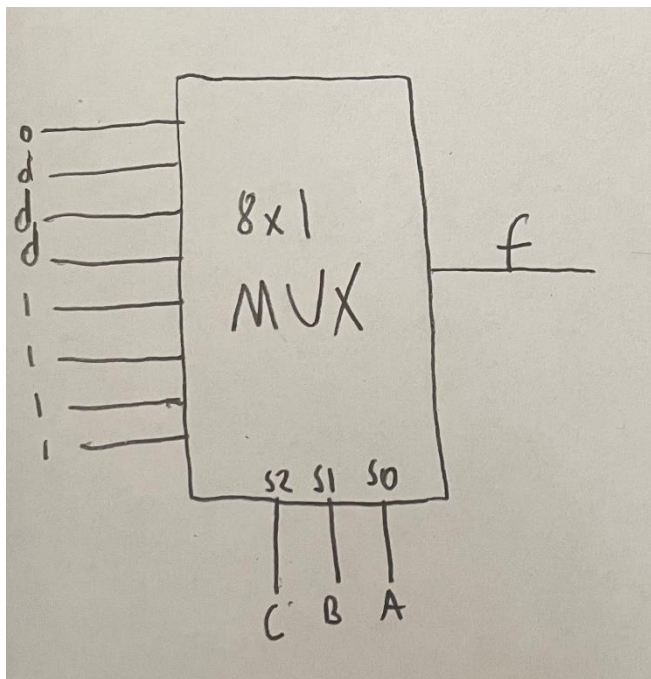
1  `timescale 1ns / 1ps
2
3  module mux8_to_1( input [2:0]s, [7:0]in, output output1);
4
5      logic [1:0]output2;
6      logic s2, d0, d1;
7
8      mux4_to_2 mux1(1,s[1:0],in[3:0], output2[0]);
9      mux4_to_2 mux2(1,s[1:0],in[7:4], output2[1]);
10
11     not( s2, s[2]);
12
13     and(d0,s2,output2[0]);
14     and(d1,s[2],output2[1]);
15
16     or(output1,d0,d1);
17
18 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:

$F(A,B,C,D) = \{ 1 \text{ if } 10 < ABCD + DABC < 200 \text{ otherwise} \}$

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 $10 < ABCD+DABC < 20$.



```

1  | `timescale 1ns / 1ps
2  |
3  | module functionF( input a,b,c,d, output out);
4  |
5  |     logic [2:0]s;
6  |     logic [7:0]in;
7  |
8  |     assign in[0] = 0;
9  |     assign in[1] = d;
10 |     assign in[2] = d;
11 |     assign in[3] = d;
12 |     assign in[4] = 1;
13 |     assign in[5] = 1;
14 |     assign in[6] = 1;
15 |     assign in[7] = 1;
16 |
17 |     assign s[2] = a;
18 |     assign s[1] = b;
19 |     assign s[0] = c;
20 |
21 |     mux8_to_1 mux(s,in,out);
22 |
23 | endmodule

```