

CS 223

Section – 6

Lab – 03

Mert Fidan

22101734

29.10.2022

- **Behavioral 1-to-2 Decoder Module:**

```
22 |
23 | module decoder12(input logic i, en, output logic y0, y1);
24 |     assign y0 = i & en;
25 |     assign y1 = ~i & en;
26 | endmodule
27 |
```

- **1-to-2 Decoder Testbench:**

```
23 | module decoder12_tb();
24 |     logic in, en, y[1:0];
25 |     decoder12 decoder(in, en, y[0], y[1]);
26 |     initial begin
27 |         in = 0; en = 0; #10;
28 |         in = 0; en = 1; #10;
29 |         in = 1; en = 0; #10;
30 |         in = 1; en = 1; #10;
31 |     end
32 | endmodule
```

- **Structural 2-to-4 Decoder Module:**

```
22 |
23 | module decoder2to4(input logic in[1:0], output logic y[3:0]);
24 |     logic out1, out2;
25 |     logic firstEnable = 1;
26 |     decoder12 first(in[0], firstEnable, out1, out2);
27 |     decoder12 second(in[1], out1, y[0], y[1]);
28 |     decoder12 third(in[1], out2, y[2], y[3]);
29 | endmodule
30 |
```

- **2-to-4 Decoder Testbench:**

```
23 | module decoder24_tb();
24 |     logic in[1:0], out[3:0];
25 |     decoder2to4 decoder(in[0], in[1], out[0], out[1], out[2], out[3]);
26 |     initial begin
27 |         in[0] = 0; in[1] = 0; #10;
28 |         in[0] = 0; in[1] = 1; #10;
29 |         in[0] = 1; in[1] = 0; #10;
30 |         in[0] = 1; in[1] = 1; #10;
31 |     end
32 | endmodule
```

- **Behavioral 2-to-1 Multiplexer Module:**

```
22 |
23 | module mux2l(input logic n1, n2, s, output logic res);
24 |     assign res = (~s & n1) + (s & n2);
25 | endmodule
26 |
```

- **2-to-1 Multiplexer Testbench:**

```
22 |
23 | module mux2l_tb();
24 |     logic n1, n2, s, res;
25 |     mux2l mux(n1, n2, s, res);
26 |     initial begin
27 |         n1 = 0; n2 = 0; s = 0; #10;
28 |         n1 = 0; n2 = 0; s = 1; #10;
29 |         n1 = 0; n2 = 1; s = 0; #10;
30 |         n1 = 0; n2 = 1; s = 1; #10;
31 |         n1 = 1; n2 = 0; s = 0; #10;
32 |         n1 = 1; n2 = 0; s = 1; #10;
33 |         n1 = 1; n2 = 1; s = 0; #10;
34 |         n1 = 1; n2 = 1; s = 1; #10;
35 |     end
36 | endmodule
```

- **Structural 4-to-1 Multiplexer Module:**

```
22 |
23 | module mux4l(input logic d1, d2, d3, d4, s1, s2, output logic y);
24 |     logic out1, out2;
25 |     mux2l first(d1, d2, s1, out1);
26 |     mux2l second(d3, d4, s1, out2);
27 |     mux2l third(out1, out2, s2, y);
28 | endmodule
```

• 4-to-1 Multiplexer Testbench:

```

22
23 module mux4l_tb();
24     logic d[3:0], s[1:0], y;
25     mux4l mux4tol(d[0], d[1], d[2], d[3], s[0], s[1], y);
26     initial begin
27         d[0] = 0; d[1] = 0; d[2] = 0; d[3] = 0; s[0] = 0; s[1] = 0; #10;
28         d[0] = 0; d[1] = 0; d[2] = 0; d[3] = 1; s[0] = 0; s[1] = 0; #10;
29         d[0] = 0; d[1] = 0; d[2] = 1; d[3] = 0; s[0] = 0; s[1] = 0; #10;
30         d[0] = 0; d[1] = 1; d[2] = 0; d[3] = 0; s[0] = 0; s[1] = 0; #10;
31         d[0] = 0; d[1] = 0; d[2] = 1; d[3] = 1; s[0] = 0; s[1] = 0; #10;
32         d[0] = 0; d[1] = 1; d[2] = 0; d[3] = 1; s[0] = 0; s[1] = 0; #10;
33         d[0] = 0; d[1] = 1; d[2] = 1; d[3] = 0; s[0] = 0; s[1] = 0; #10;
34         d[0] = 0; d[1] = 1; d[2] = 1; d[3] = 1; s[0] = 0; s[1] = 0; #10;
35         d[0] = 1; d[1] = 0; d[2] = 0; d[3] = 0; s[0] = 0; s[1] = 0; #10;
36         d[0] = 1; d[1] = 0; d[2] = 0; d[3] = 1; s[0] = 0; s[1] = 0; #10;
37         d[0] = 1; d[1] = 0; d[2] = 1; d[3] = 0; s[0] = 0; s[1] = 0; #10;
38         d[0] = 1; d[1] = 1; d[2] = 0; d[3] = 0; s[0] = 0; s[1] = 0; #10;
39         d[0] = 1; d[1] = 0; d[2] = 1; d[3] = 1; s[0] = 0; s[1] = 0; #10;
40         d[0] = 1; d[1] = 1; d[2] = 0; d[3] = 1; s[0] = 0; s[1] = 0; #10;
41         d[0] = 1; d[1] = 1; d[2] = 1; d[3] = 0; s[0] = 0; s[1] = 0; #10;
42         d[0] = 1; d[1] = 1; d[2] = 1; d[3] = 1; s[0] = 0; s[1] = 0; #10;
43         d[0] = 0; d[1] = 0; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 1; #10;
44         d[0] = 0; d[1] = 0; d[2] = 0; d[3] = 1; s[0] = 1; s[1] = 1; #10;
45         d[0] = 0; d[1] = 0; d[2] = 1; d[3] = 0; s[0] = 1; s[1] = 1; #10;
46         d[0] = 0; d[1] = 1; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 1; #10;
47         d[0] = 0; d[1] = 0; d[2] = 1; d[3] = 1; s[0] = 1; s[1] = 1; #10;
48         d[0] = 0; d[1] = 1; d[2] = 0; d[3] = 1; s[0] = 1; s[1] = 1; #10;
49         d[0] = 0; d[1] = 1; d[2] = 1; d[3] = 0; s[0] = 1; s[1] = 1; #10;
50         d[0] = 0; d[1] = 1; d[2] = 1; d[3] = 1; s[0] = 1; s[1] = 1; #10;
51         d[0] = 1; d[1] = 0; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 1; #10;
52         d[0] = 1; d[1] = 0; d[2] = 0; d[3] = 1; s[0] = 1; s[1] = 1; #10;
53         d[0] = 1; d[1] = 0; d[2] = 1; d[3] = 0; s[0] = 1; s[1] = 1; #10;
54         d[0] = 1; d[1] = 1; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 1; #10;
55         d[0] = 1; d[1] = 0; d[2] = 1; d[3] = 1; s[0] = 0; s[1] = 1; #10;
56         d[0] = 1; d[1] = 1; d[2] = 0; d[3] = 1; s[0] = 0; s[1] = 1; #10;
57         d[0] = 1; d[1] = 1; d[2] = 1; d[3] = 0; s[0] = 0; s[1] = 1; #10;
58         d[0] = 1; d[1] = 1; d[2] = 1; d[3] = 1; s[0] = 0; s[1] = 1; #10;
59         d[0] = 0; d[1] = 0; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 0; #10;
60         d[0] = 0; d[1] = 0; d[2] = 0; d[3] = 1; s[0] = 1; s[1] = 0; #10;
61         d[0] = 0; d[1] = 0; d[2] = 1; d[3] = 0; s[0] = 1; s[1] = 0; #10;
62         d[0] = 0; d[1] = 1; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 0; #10;
63         d[0] = 0; d[1] = 0; d[2] = 1; d[3] = 1; s[0] = 1; s[1] = 0; #10;
64         d[0] = 0; d[1] = 1; d[2] = 0; d[3] = 1; s[0] = 1; s[1] = 0; #10;
65         d[0] = 0; d[1] = 1; d[2] = 1; d[3] = 0; s[0] = 1; s[1] = 0; #10;
66         d[0] = 0; d[1] = 1; d[2] = 1; d[3] = 1; s[0] = 1; s[1] = 0; #10;
67         d[0] = 1; d[1] = 0; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 0; #10;
68         d[0] = 1; d[1] = 0; d[2] = 0; d[3] = 1; s[0] = 1; s[1] = 0; #10;
69         d[0] = 1; d[1] = 0; d[2] = 1; d[3] = 0; s[0] = 1; s[1] = 0; #10;
70         d[0] = 1; d[1] = 1; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 0; #10;
71         d[0] = 1; d[1] = 0; d[2] = 1; d[3] = 1; s[0] = 1; s[1] = 0; #10;
72         d[0] = 1; d[1] = 1; d[2] = 0; d[3] = 1; s[0] = 1; s[1] = 0; #10;
73         d[0] = 1; d[1] = 1; d[2] = 1; d[3] = 0; s[0] = 1; s[1] = 0; #10;
74         d[0] = 1; d[1] = 1; d[2] = 1; d[3] = 1; s[0] = 1; s[1] = 0; #10;
75         d[0] = 0; d[1] = 0; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 1; #10;
76         d[0] = 0; d[1] = 0; d[2] = 0; d[3] = 1; s[0] = 1; s[1] = 1; #10;
77         d[0] = 0; d[1] = 0; d[2] = 1; d[3] = 0; s[0] = 1; s[1] = 1; #10;
78         d[0] = 0; d[1] = 1; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 1; #10;
79         d[0] = 0; d[1] = 0; d[2] = 1; d[3] = 1; s[0] = 1; s[1] = 1; #10;
80         d[0] = 0; d[1] = 1; d[2] = 0; d[3] = 1; s[0] = 1; s[1] = 1; #10;
81         d[0] = 0; d[1] = 1; d[2] = 1; d[3] = 0; s[0] = 1; s[1] = 1; #10;
82         d[0] = 0; d[1] = 1; d[2] = 1; d[3] = 1; s[0] = 1; s[1] = 1; #10;
83         d[0] = 1; d[1] = 0; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 1; #10;
84         d[0] = 1; d[1] = 0; d[2] = 0; d[3] = 1; s[0] = 1; s[1] = 1; #10;
85         d[0] = 1; d[1] = 0; d[2] = 1; d[3] = 0; s[0] = 1; s[1] = 1; #10;
86         d[0] = 1; d[1] = 1; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 1; #10;
87         d[0] = 1; d[1] = 0; d[2] = 1; d[3] = 1; s[0] = 1; s[1] = 1; #10;
88         d[0] = 1; d[1] = 1; d[2] = 0; d[3] = 1; s[0] = 1; s[1] = 1; #10;
89         d[0] = 1; d[1] = 1; d[2] = 1; d[3] = 0; s[0] = 1; s[1] = 1; #10;
90         d[0] = 1; d[1] = 1; d[2] = 1; d[3] = 1; s[0] = 1; s[1] = 1; #10;
91     end
92 endmodule

```

• Structural 8-to-1 Multiplexer Module:

```

22
23 module mux8l(input logic in[7:0], sel[2:0], output logic y);
24     logic muxRes1, muxRes2, andRes1, andRes2;
25     mux4l firstMux(in[0], in[1], in[2], in[3], sel[0], sel[1], muxRes1);
26     mux4l secondMux(in[4], in[5], in[6], in[7], sel[0], sel[1], muxRes2);
27     assign andRes1 = muxRes1 & ~sel[2];
28     assign andRes2 = muxRes2 & sel[2];
29     assign y = andRes1 | andRes2;
30 endmodule

```

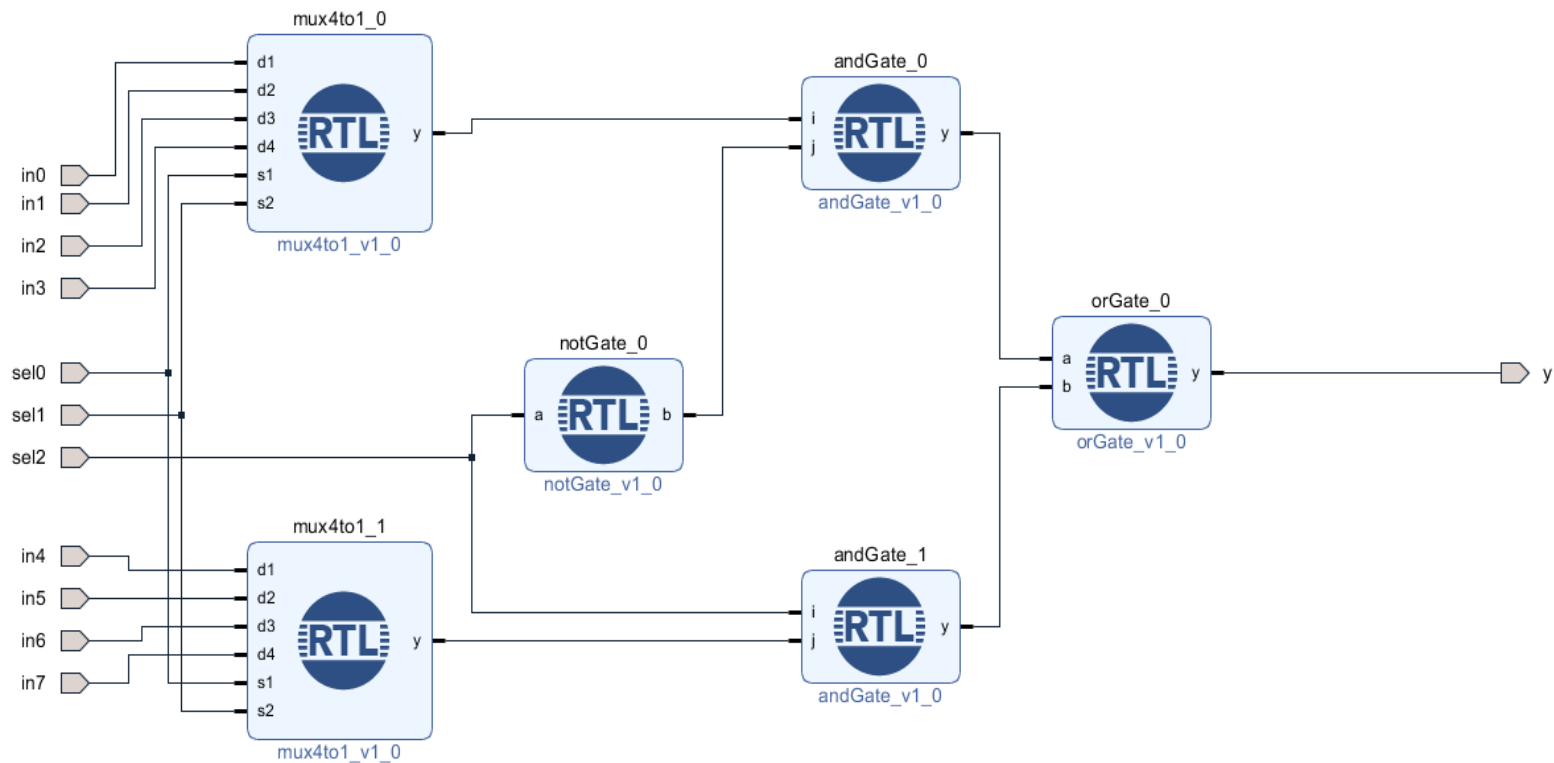
• 8-to-1 Multiplexer Testbench:

```

22
23 module mux8l_tb();
24     logic in[7:0], sel[2:0], y;
25     mux8l mux8l(in[7:0], sel[2:0], y);
26     initial begin
27         for(int j = 0; j < 8; j = j+1)
28             begin
29                 {sel[0], sel[1], sel[2]} = j;
30                 for(int i = 0; i < 256; i = i+1)
31                     begin
32                         #0.25;
33                         {in[0], in[1], in[2], in[3], in[4], in[5], in[6], in[7]} = i;
34                     end
35                 #0.25;
36             end
37     end
38 endmodule

```

• 8-to-1 Multiplexer Block Design:



• Lab Function Module:

```

22  |
23  | module labFunction(input logic a, b, c, d, output logic res);
24  |     mux81 mux(b, b, b, b, ~c, ~c, ~c, ~c, a, c, d, res);
25  | endmodule
26  |

```

• Lab Function Block Design:

