CS 223

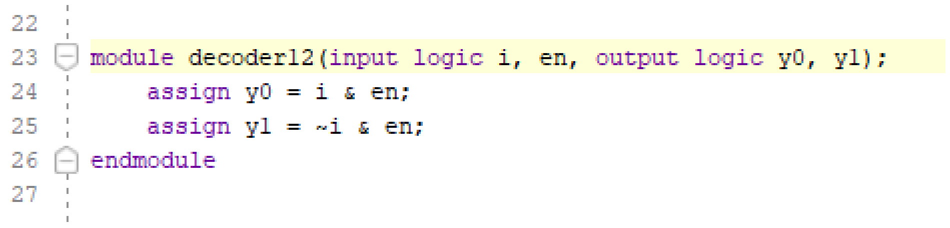
Section – 6

Lab – 03

Mert Fidan
22101734

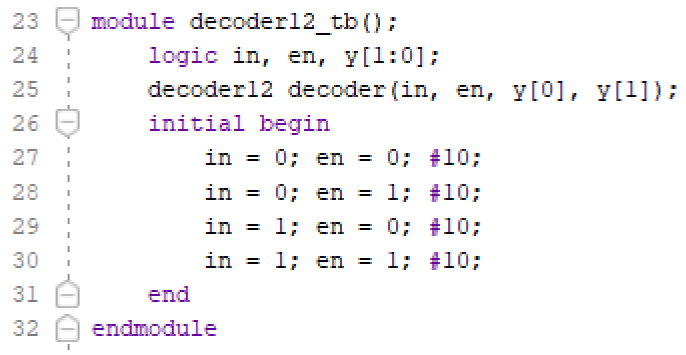
29.10.2022

- **<u>Behavioral 1-to-2 Decoder Module:</u>**



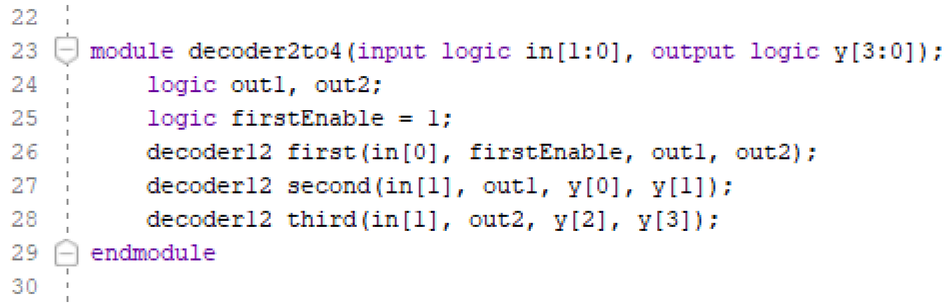
```

module decoder12(input logic i, en, output logic y0, y1);
    assign y0 = i & en;
    assign y1 = ~i & en;
endmodule

```

- **<u>1-to-2 Decoder Testbench:</u>**



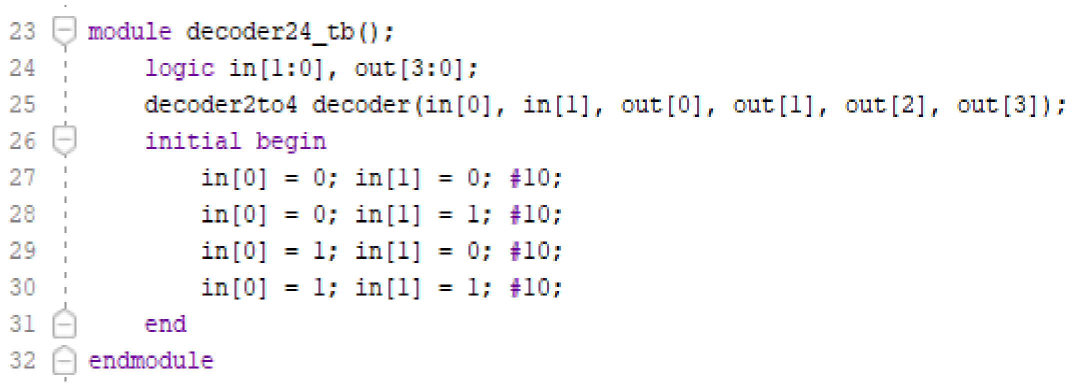
```
module decoder12_tb();
    logic in, en, y[1:0];
    decoder12 decoder(in, en, y[0], y[1]);
    initial begin
        in = 0; en = 0; #10;
        in = 0; en = 1; #10;
        in = 1; en = 0; #10;
        in = 1; en = 1; #10;
    end
endmodule
```

- **<u>Structural 2-to-4 Decoder Module:</u>**



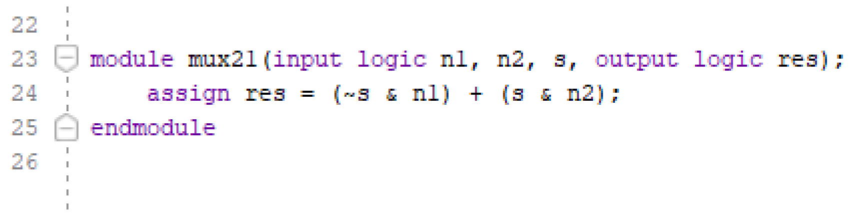
```

module decoder2to4(input logic in[1:0], output logic y[3:0]);
    logic out1, out2;
    logic firstEnable = 1;
    decoder12 first(in[0], firstEnable, out1, out2);
    decoder12 second(in[1], out1, y[0], y[1]);
    decoder12 third(in[1], out2, y[2], y[3]);
endmodule

```

- **<u>2-to-4 Decoder Testbench:</u>**



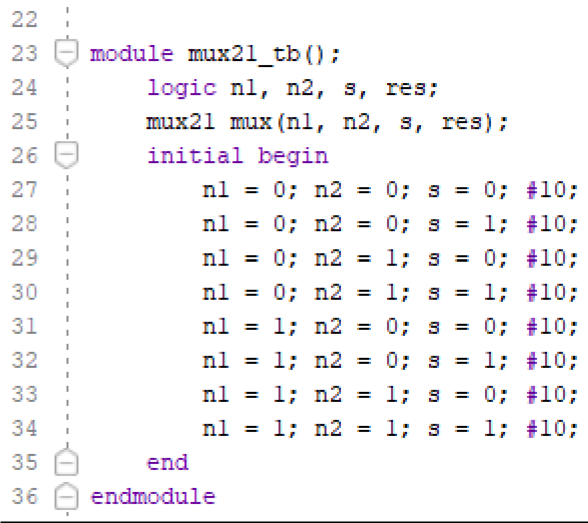
```
module decoder24_tb();
    logic in[1:0], out[3:0];
    decoder2to4 decoder(in[0], in[1], out[0], out[1], out[2], out[3]);
    initial begin
        in[0] = 0; in[1] = 0; #10;
        in[0] = 0; in[1] = 1; #10;
        in[0] = 1; in[1] = 0; #10;
        in[0] = 1; in[1] = 1; #10;
    end
endmodule
```

- <u>**Behavioral 2-to-1 Multiplexer Module:**</u>



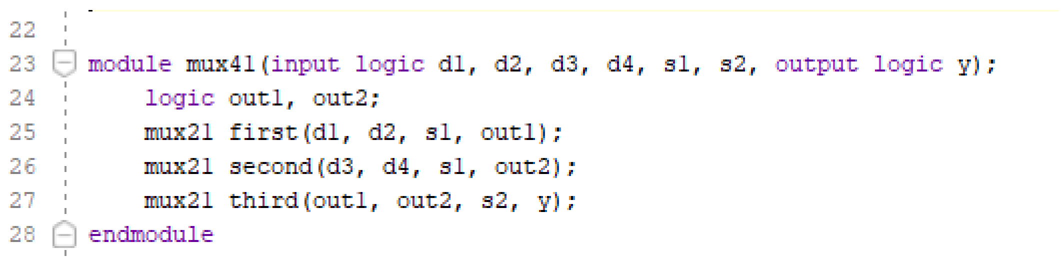
```
module mux21(input logic n1, n2, s, output logic res);
    assign res = (~s & n1) + (s & n2);
endmodule
```

- <u>**2-to-1 Multiplexer Testbench:**</u>



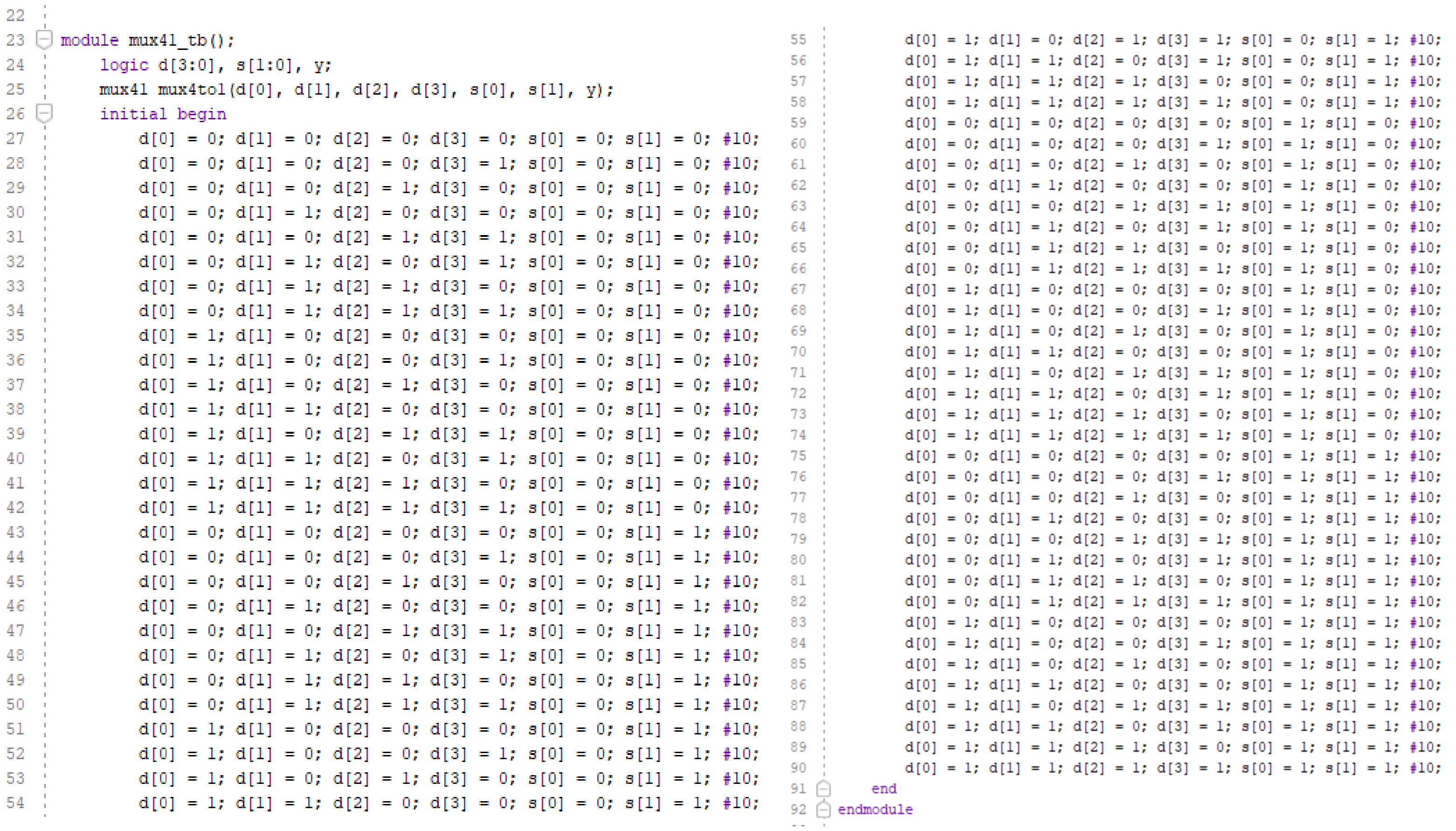
```
module mux21_tb();
    logic n1, n2, s, res;
    mux21 mux(n1, n2, s, res);
    initial begin
        n1 = 0; n2 = 0; s = 0; #10;
        n1 = 0; n2 = 0; s = 1; #10;
        n1 = 0; n2 = 1; s = 0; #10;
        n1 = 0; n2 = 1; s = 1; #10;
        n1 = 1; n2 = 0; s = 0; #10;
        n1 = 1; n2 = 0; s = 1; #10;
        n1 = 1; n2 = 1; s = 0; #10;
        n1 = 1; n2 = 1; s = 1; #10;
    end
endmodule
```

- <u>**Structural 4-to-1 Multiplexer Module:**</u>



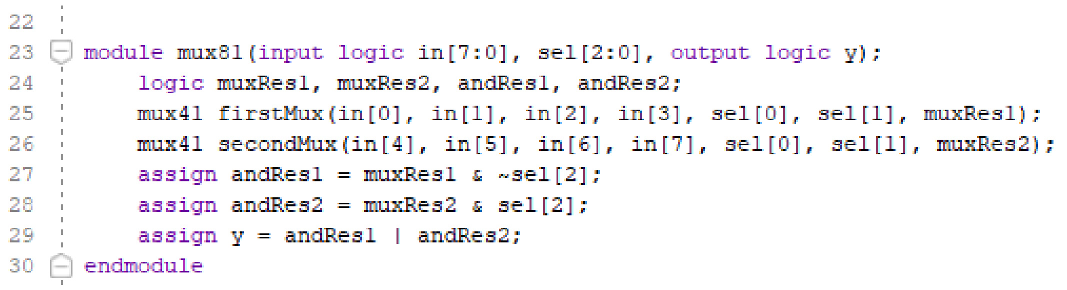
```
module mux41(input logic d1, d2, d3, d4, s1, s2, output logic y);
    logic out1, out2;
    mux21 first(d1, d2, s1, out1);
    mux21 second(d3, d4, s1, out2);
    mux21 third(out1, out2, s2, y);
endmodule
```

- **4-to-1 Multiplexer Testbench:**



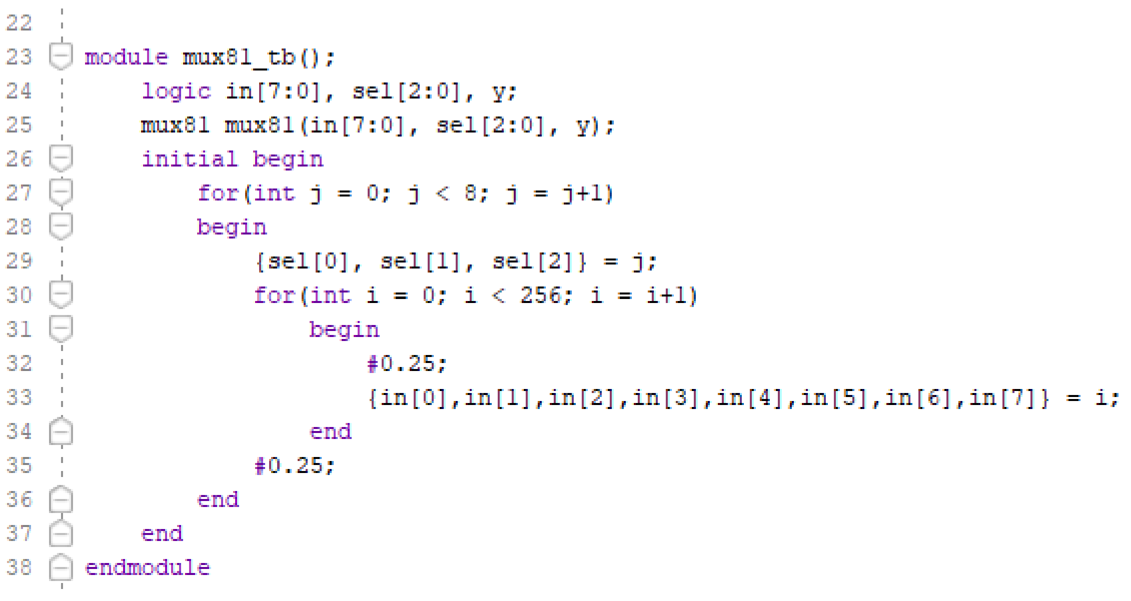
```
module mux41_tb();
    logic d[3:0], s[1:0], y;
    mux41 mux4to1(d[0], d[1], d[2], d[3], s[0], s[1], y);
    initial begin
        d[0] = 0; d[1] = 0; d[2] = 0; d[3] = 0; s[0] = 0; s[1] = 0; #10;
        d[0] = 0; d[1] = 0; d[2] = 0; d[3] = 1; s[0] = 0; s[1] = 0; #10;
        d[0] = 0; d[1] = 0; d[2] = 1; d[3] = 0; s[0] = 0; s[1] = 0; #10;
        d[0] = 0; d[1] = 1; d[2] = 0; d[3] = 0; s[0] = 0; s[1] = 0; #10;
        d[0] = 0; d[1] = 0; d[2] = 1; d[3] = 1; s[0] = 0; s[1] = 0; #10;
        d[0] = 0; d[1] = 1; d[2] = 0; d[3] = 1; s[0] = 0; s[1] = 0; #10;
        d[0] = 0; d[1] = 1; d[2] = 1; d[3] = 0; s[0] = 0; s[1] = 0; #10;
        d[0] = 0; d[1] = 1; d[2] = 1; d[3] = 1; s[0] = 0; s[1] = 0; #10;
        d[0] = 1; d[1] = 0; d[2] = 0; d[3] = 0; s[0] = 0; s[1] = 0; #10;
        d[0] = 1; d[1] = 0; d[2] = 0; d[3] = 1; s[0] = 0; s[1] = 0; #10;
        d[0] = 1; d[1] = 0; d[2] = 1; d[3] = 0; s[0] = 0; s[1] = 0; #10;
        d[0] = 1; d[1] = 1; d[2] = 0; d[3] = 0; s[0] = 0; s[1] = 0; #10;
        d[0] = 1; d[1] = 0; d[2] = 1; d[3] = 1; s[0] = 0; s[1] = 0; #10;
        d[0] = 1; d[1] = 1; d[2] = 0; d[3] = 1; s[0] = 0; s[1] = 0; #10;
        d[0] = 1; d[1] = 1; d[2] = 1; d[3] = 0; s[0] = 0; s[1] = 0; #10;
        d[0] = 1; d[1] = 1; d[2] = 1; d[3] = 1; s[0] = 0; s[1] = 0; #10;
        d[0] = 0; d[1] = 0; d[2] = 0; d[3] = 0; s[0] = 0; s[1] = 1; #10;
        d[0] = 0; d[1] = 0; d[2] = 0; d[3] = 1; s[0] = 0; s[1] = 1; #10;
        d[0] = 0; d[1] = 0; d[2] = 1; d[3] = 0; s[0] = 0; s[1] = 1; #10;
        d[0] = 0; d[1] = 1; d[2] = 0; d[3] = 0; s[0] = 0; s[1] = 1; #10;
        d[0] = 0; d[1] = 0; d[2] = 1; d[3] = 1; s[0] = 0; s[1] = 1; #10;
        d[0] = 0; d[1] = 1; d[2] = 0; d[3] = 1; s[0] = 0; s[1] = 1; #10;
        d[0] = 0; d[1] = 1; d[2] = 1; d[3] = 0; s[0] = 0; s[1] = 1; #10;
        d[0] = 0; d[1] = 1; d[2] = 1; d[3] = 1; s[0] = 0; s[1] = 1; #10;
        d[0] = 1; d[1] = 0; d[2] = 0; d[3] = 0; s[0] = 0; s[1] = 1; #10;
        d[0] = 1; d[1] = 0; d[2] = 0; d[3] = 1; s[0] = 0; s[1] = 1; #10;
        d[0] = 1; d[1] = 0; d[2] = 1; d[3] = 0; s[0] = 0; s[1] = 1; #10;
        d[0] = 1; d[1] = 1; d[2] = 0; d[3] = 0; s[0] = 0; s[1] = 1; #10;
        d[0] = 1; d[1] = 0; d[2] = 1; d[3] = 1; s[0] = 0; s[1] = 1; #10;
        d[0] = 1; d[1] = 1; d[2] = 0; d[3] = 1; s[0] = 0; s[1] = 1; #10;
        d[0] = 1; d[1] = 1; d[2] = 1; d[3] = 0; s[0] = 0; s[1] = 1; #10;
        d[0] = 1; d[1] = 1; d[2] = 1; d[3] = 1; s[0] = 0; s[1] = 1; #10;
        d[0] = 0; d[1] = 0; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 0; #10;
        d[0] = 0; d[1] = 0; d[2] = 0; d[3] = 1; s[0] = 1; s[1] = 0; #10;
        d[0] = 0; d[1] = 0; d[2] = 1; d[3] = 0; s[0] = 1; s[1] = 0; #10;
        d[0] = 0; d[1] = 1; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 0; #10;
        d[0] = 0; d[1] = 0; d[2] = 1; d[3] = 1; s[0] = 1; s[1] = 0; #10;
        d[0] = 0; d[1] = 1; d[2] = 0; d[3] = 1; s[0] = 1; s[1] = 0; #10;
        d[0] = 0; d[1] = 1; d[2] = 1; d[3] = 0; s[0] = 1; s[1] = 0; #10;
        d[0] = 0; d[1] = 1; d[2] = 1; d[3] = 1; s[0] = 1; s[1] = 0; #10;
        d[0] = 1; d[1] = 0; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 0; #10;
        d[0] = 1; d[1] = 0; d[2] = 0; d[3] = 1; s[0] = 1; s[1] = 0; #10;
        d[0] = 1; d[1] = 0; d[2] = 1; d[3] = 0; s[0] = 1; s[1] = 0; #10;
        d[0] = 1; d[1] = 1; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 0; #10;
        d[0] = 1; d[1] = 0; d[2] = 1; d[3] = 1; s[0] = 1; s[1] = 0; #10;
        d[0] = 1; d[1] = 1; d[2] = 0; d[3] = 1; s[0] = 1; s[1] = 0; #10;
        d[0] = 1; d[1] = 1; d[2] = 1; d[3] = 0; s[0] = 1; s[1] = 0; #10;
        d[0] = 1; d[1] = 1; d[2] = 1; d[3] = 1; s[0] = 1; s[1] = 0; #10;
        d[0] = 0; d[1] = 0; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 1; #10;
        d[0] = 0; d[1] = 0; d[2] = 0; d[3] = 1; s[0] = 1; s[1] = 1; #10;
        d[0] = 0; d[1] = 0; d[2] = 1; d[3] = 0; s[0] = 1; s[1] = 1; #10;
        d[0] = 0; d[1] = 1; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 1; #10;
        d[0] = 0; d[1] = 0; d[2] = 1; d[3] = 1; s[0] = 1; s[1] = 1; #10;
        d[0] = 0; d[1] = 1; d[2] = 0; d[3] = 1; s[0] = 1; s[1] = 1; #10;
        d[0] = 0; d[1] = 1; d[2] = 1; d[3] = 0; s[0] = 1; s[1] = 1; #10;
        d[0] = 0; d[1] = 1; d[2] = 1; d[3] = 1; s[0] = 1; s[1] = 1; #10;
        d[0] = 1; d[1] = 0; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 1; #10;
        d[0] = 1; d[1] = 0; d[2] = 0; d[3] = 1; s[0] = 1; s[1] = 1; #10;
        d[0] = 1; d[1] = 0; d[2] = 1; d[3] = 0; s[0] = 1; s[1] = 1; #10;
        d[0] = 1; d[1] = 1; d[2] = 0; d[3] = 0; s[0] = 1; s[1] = 1; #10;
        d[0] = 1; d[1] = 0; d[2] = 1; d[3] = 1; s[0] = 1; s[1] = 1; #10;
        d[0] = 1; d[1] = 1; d[2] = 0; d[3] = 1; s[0] = 1; s[1] = 1; #10;
        d[0] = 1; d[1] = 1; d[2] = 1; d[3] = 0; s[0] = 1; s[1] = 1; #10;
        d[0] = 1; d[1] = 1; d[2] = 1; d[3] = 1; s[0] = 1; s[1] = 1; #10;
    end
endmodule
```

- **Structural 8-to-1 Multiplexer Module:**



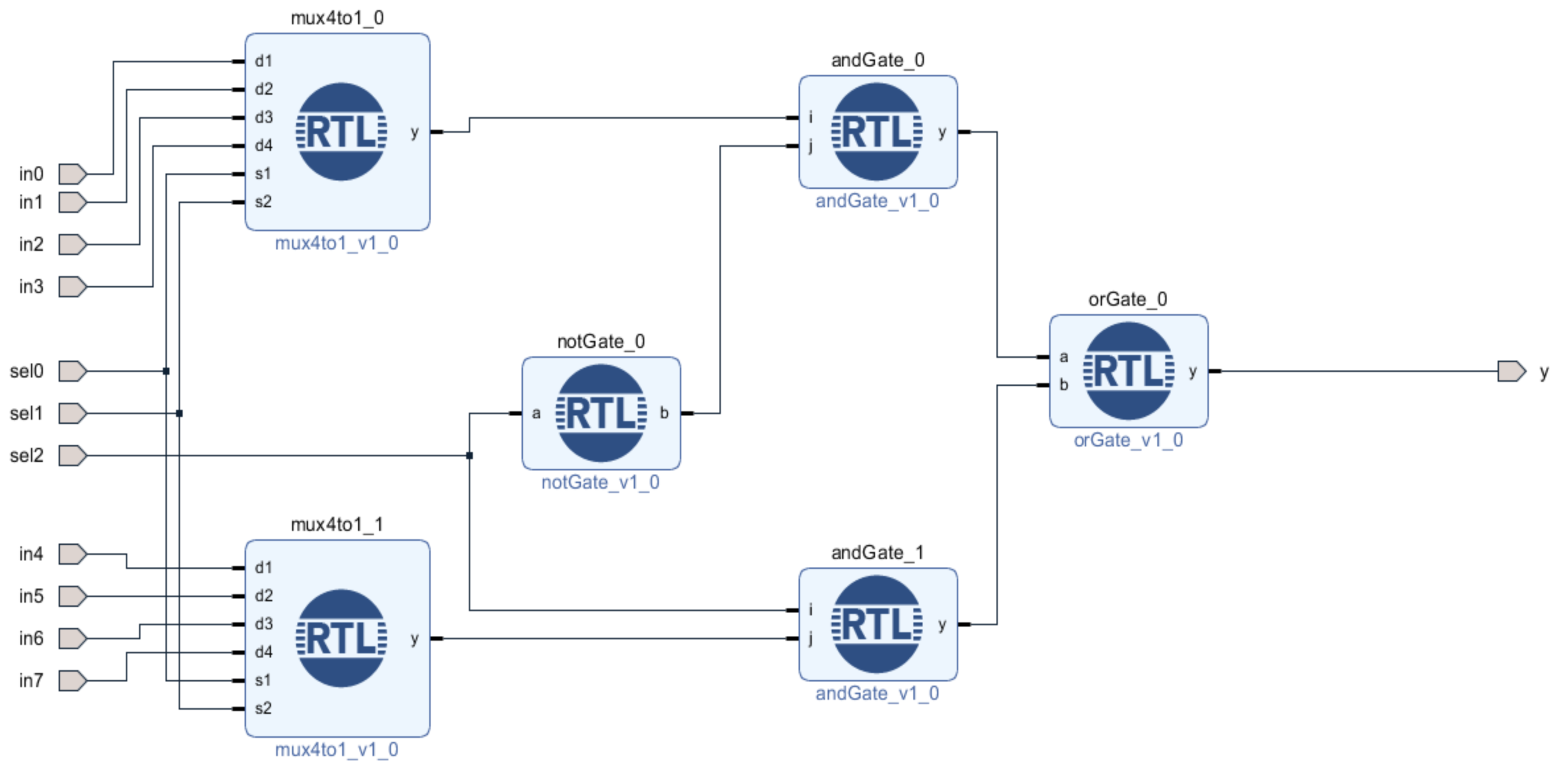
```
module mux81(input logic in[7:0], sel[2:0], output logic y);
    logic muxRes1, muxRes2, andRes1, andRes2;
    mux41 firstMux(in[0], in[1], in[2], in[3], sel[0], sel[1], muxRes1);
    mux41 secondMux(in[4], in[5], in[6], in[7], sel[0], sel[1], muxRes2);
    assign andRes1 = muxRes1 & ~sel[2];
    assign andRes2 = muxRes2 & sel[2];
    assign y = andRes1 | andRes2;
endmodule
```

- **8-to-1 Multiplexer Testbench:**

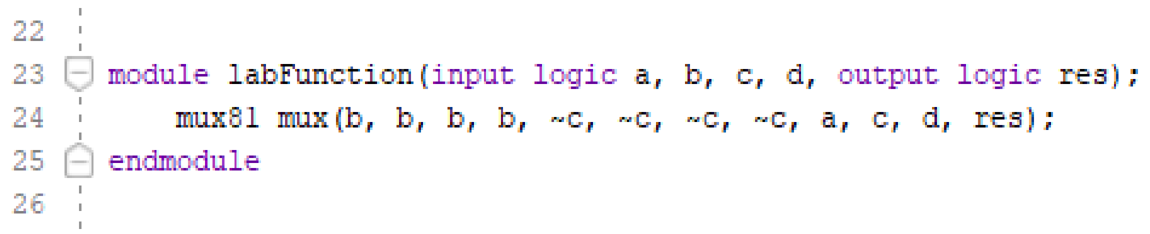


```
module mux81_tb();
    logic in[7:0], sel[2:0], y;
    mux81 mux81(in[7:0], sel[2:0], y);
    initial begin
        for(int j = 0; j < 8; j = j+1)
        begin
            {sel[0], sel[1], sel[2]} = j;
            for(int i = 0; i < 256; i = i+1)
            begin
                #0.25;
                {in[0],in[1],in[2],in[3],in[4],in[5],in[6],in[7]} = i;
            end
            #0.25;
        end
    end
endmodule
```

- **8-to-1 Multiplexer Block Design:**

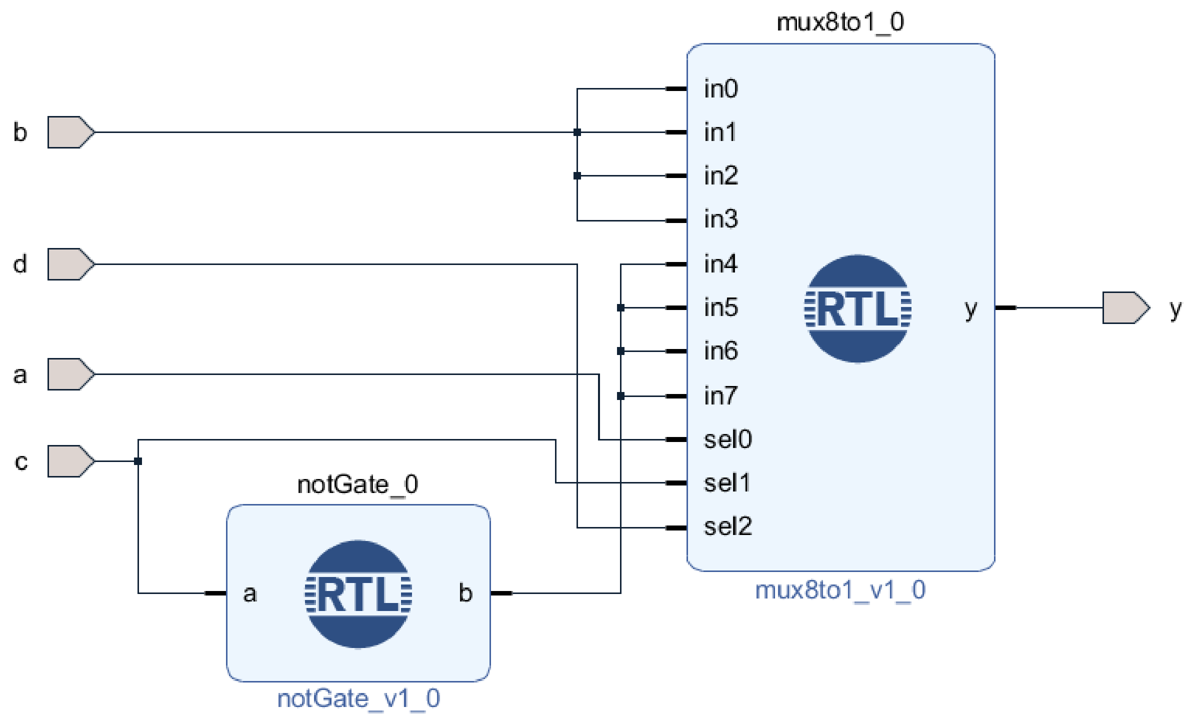


- **Lab Function Module:**



```
module labFunction(input logic a, b, c, d, output logic res);
    mux81 mux(b, b, b, b, ~c, ~c, ~c, ~c, a, c, d, res);
endmodule
```

- **Lab Function Block Design:**


mux8to1_0
b
in0
in1
in2
in3
d
in4
in5
in6
in7
sel0
sel1
sel2
y
y
a
c
notGate_0
a
b
notGate_v1_0
mux8to1_v1_0