CS 223

Digital Design

Section 01

Laboratory Assignment 3

Modeling Decoders and MUXs in System Verilog

Arda Önal

21903350

24.10.2020

b) Behavioral SystemVerilog module for 2-to-4 decoder and a testbench for it.

```
module two_to_four_decoder( input logic a0, a1, output logic y0, y1, y2, y3);

   assign y0 = ~a0 & ~a1;
   assign y1 = a0  & ~a1;
   assign y2 = ~a0 & a1;
   assign y3 = a0 & a1;

endmodule
```

Testbench:

```
module testbench_two_to_four_decoder();
   logic a0, a1;
   logic y0, y1, y2, y3;

   two_to_four_decoder dut( a0, a1, y0, y1, y2, y3);

   initial begin
      a0 = 0; a1 = 0; #10;
      if ( y0 != 1 | y1 != 0 | y2 != 0 | y3 != 0) $display("00 failed.");
      a0 = 1; #10;
      if ( y0 != 0 | y1 != 1 | y2 != 0 | y3 != 0) $display("10 failed.");
      a0 = 0; a1 = 1; #10;
      if ( y0 != 0 | y1 != 0 | y2 != 1 | y3 != 0) $display("01 failed.");
      a0 = 1; #10;
      if ( y0 != 0 | y1 != 0 | y2 != 0 | y3 != 1) $display("11 failed.");
   end
endmodule
```

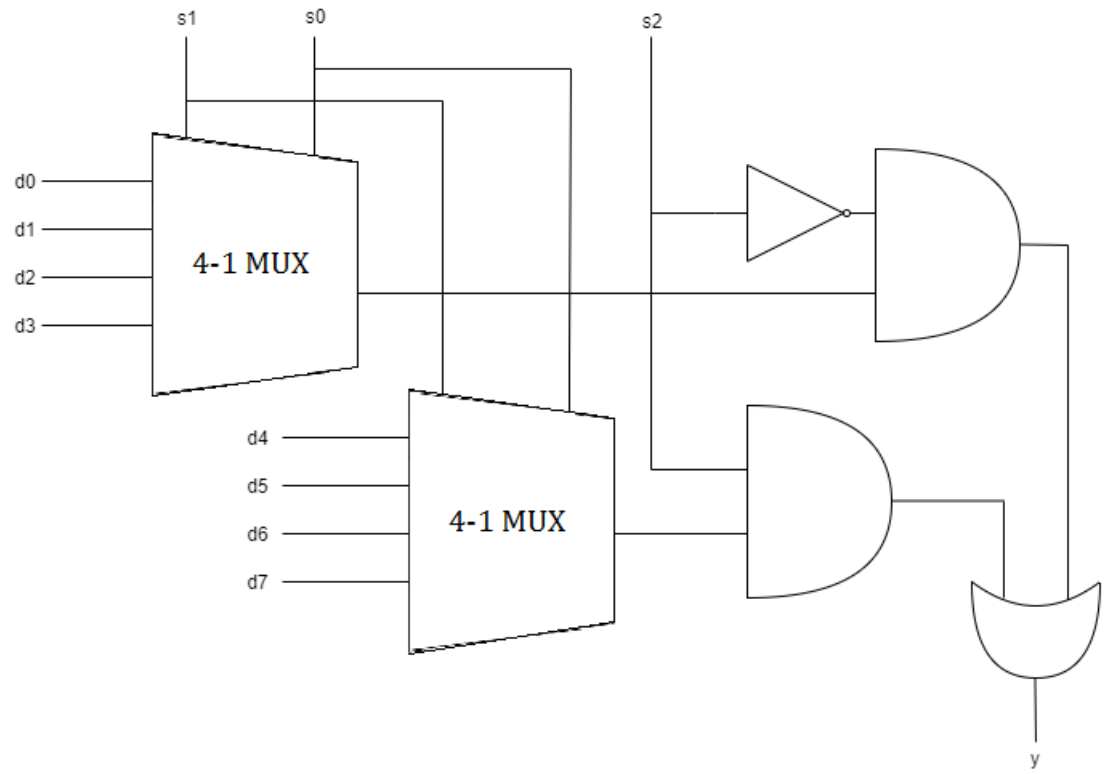
c) Behavioral SystemVerilog module for 4-to-1 multiplexer.

```
module four_to_one_mux(input logic [3:0]d,[1:0]s, output logic y);

assign y = s[1] ? (s[0] ? d[3] : d[2]) : ( s[0] ? d[1] : d[0]);

endmodule
```

Testbench:

```
module testbench_four_to_one_mux();
   logic [3:0]d;
   logic [1:0]s;
   logic y;

   four_to_one_mux dut(d,s,y);
   initial begin
      for ( int i = 0; i < 64; i++) begin
         {d, s} = i; #9;
         if ( y != d[s]) $display( "%0d failed", {d,s});
      end
   end
endmodule
```

d) Schematic (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. Prepare a test bench for it.



Structural System Verilog module of 8-to-1 MUX by using two 4-to-1 MUX modules:

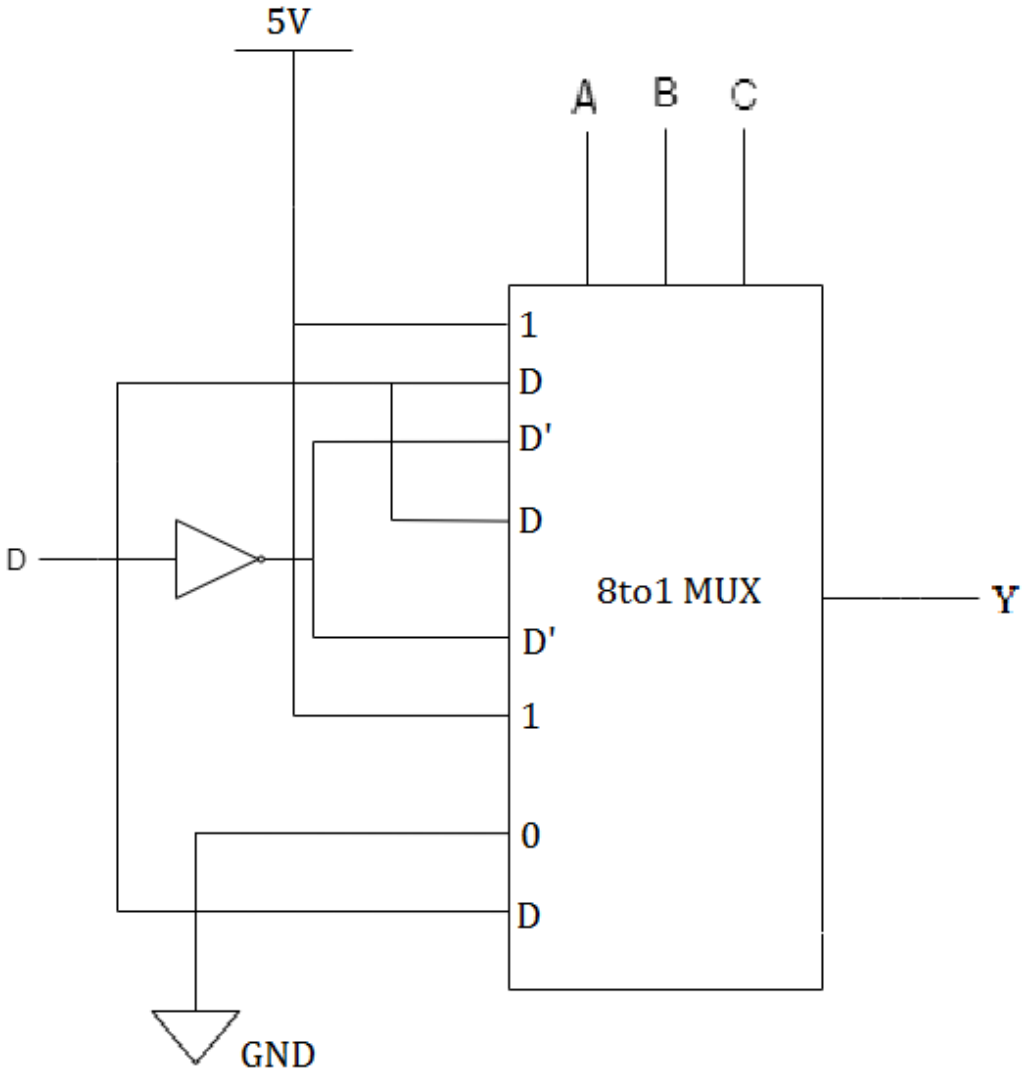
```
module eight_to_one_mux(input logic [7:0]d,[2:0]s, output logic y);
    logic mux0;
    logic mux1;
    logic node0;
    logic node1;

    four_to_one_mux multiplexer0( d[3:0], s[1:0], mux0);
    four_to_one_mux multiplexer1( d[7:4], s[1:0], mux1);
```

```
    and( node0, ~s[2], mux0);
    and( node1, s[2], mux1);
    or( y, node0, node1);
endmodule
```

Testbench:

```
module testbench_eight_to_one_mux();

    logic [7:0]d;
    logic [2:0]s;
    logic y;

    eight_to_one_mux mux8(d[7:0],s[2:0],y);

    initial begin
        for ( int i = 'd0; i < 'd2048; i++) begin
            {d,s} = i; #0.3;
            if ( y != d[s]) $display( "%0d failed", {d,s});
            end
    end
endmodule
```

e) Schematic (block diagram) and SystemVerilog module for F(A,B,C,D)=∑(0,1,3,4,7,8,10,11,15) function, using one (not two) 8-to-1 multiplexer and an inverter.




```
module functionF(input logic a,b,c,d, output logic y);
    logic dnot;
    not inverter(dnot,d);
    eight_to_one_mux mux8({d, 1'b0, 1'b1, dnot, d, dnot, d, 1'b1}, {a, b, c}, y);
endmodule
```

```
module partE_testbench();
    logic a,b,c,d;
    logic y;

    functionF dut( a,b,c,d, y);

    initial begin

    for ( int i = 0; i < 16; i++) begin
        {a,b,c,d} = i; #10;
        end
    end
endmodule
```