CS223 DIGITAL DESIGN

SECTION 3

Laboratory Assignment 3:

Modeling Decoders and MUXs in System Verilog

Nurettin Onur Vural 21902330

25.10.2020

PRELIMINARY REPORT

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

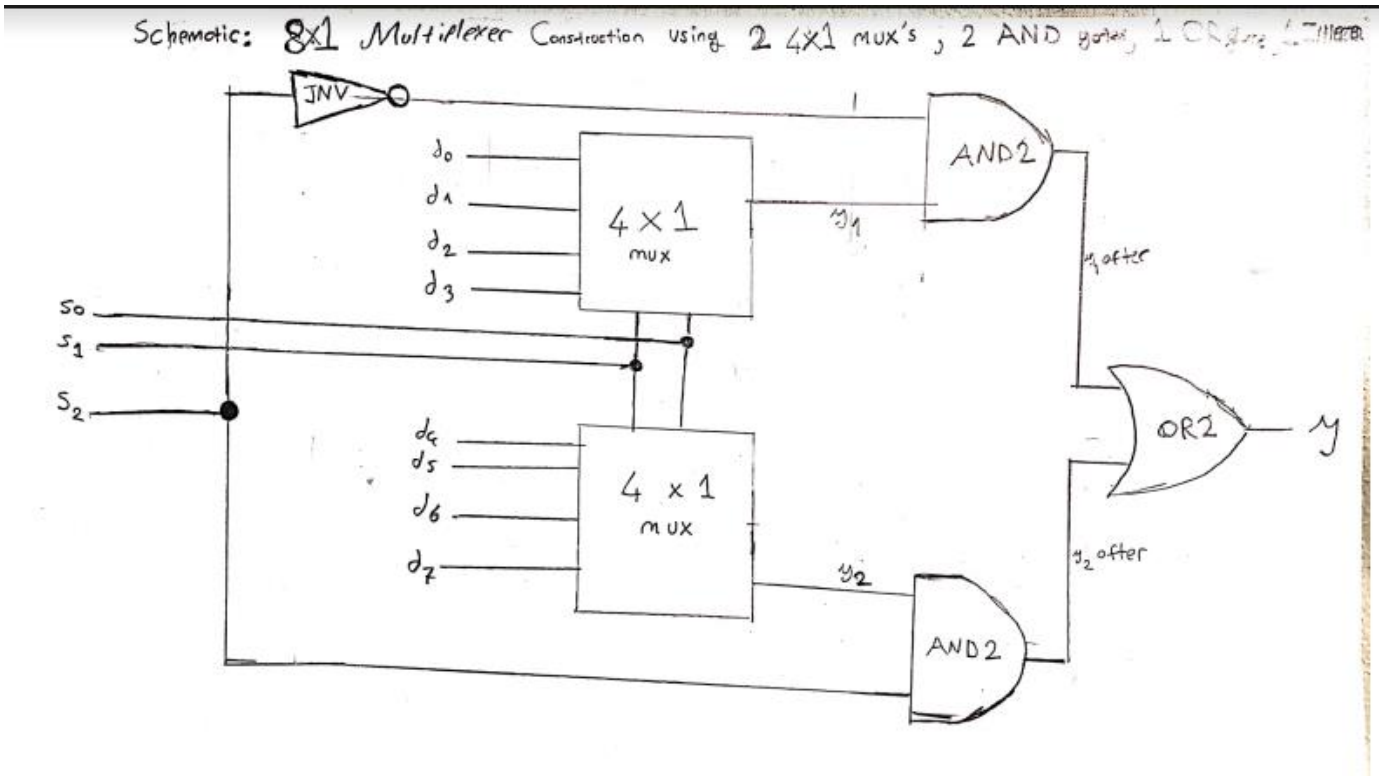
```
module twoToFourDecoder(
   input logic enable, input1, input0,
   output logic y3, y2, y1, y0
   );


  always_comb
  if (enable)
     begin
     y3 = input1 & input0;
     y2 = input1 & ~input0;
     y1 = ~input1 & input0;
     y0 = ~input1 & ~input0;
     end
  else
     begin
     y3 = 1'b0;
     y2 = 1'b0;
     y1 = 1'b0;
     y0 = 1'b0;
     end
endmodule
//TESTBENCH FOR 2-TO-4 DECODER
module twoToFourDecoder_testbench();
logic enable, input1, input0;
logic y3, y2, y1, y0;

twoToFourDecoder dut(enable, input1, input0, y3, y2, y1, y0);
 // apply inputs one at a time
 initial begin
 enable = 0; input1 = 0; input0 = 0; #10;
```

```
 input0 = 1; #10;
 input1 = 1; input0 = 0; #10;
 input0 = 1; #10;
 enable = 1; input1 = 0; input0 = 0; #10;
 input0 = 1; #10;
 input1 = 1; input0 = 0; #10;
 input0 = 1; #10;
 end
endmodule
```

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

```
module fourToOneMultiplexer(
   input logic s1, s0, d0, d1, d2, d3,
   output logic y
   );
   assign y = (~s1 & ~s0 & d0) + (~s1 & s0 & d1) + (s1 & ~s0 & d2) + (s1 & s0 & d3);
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.

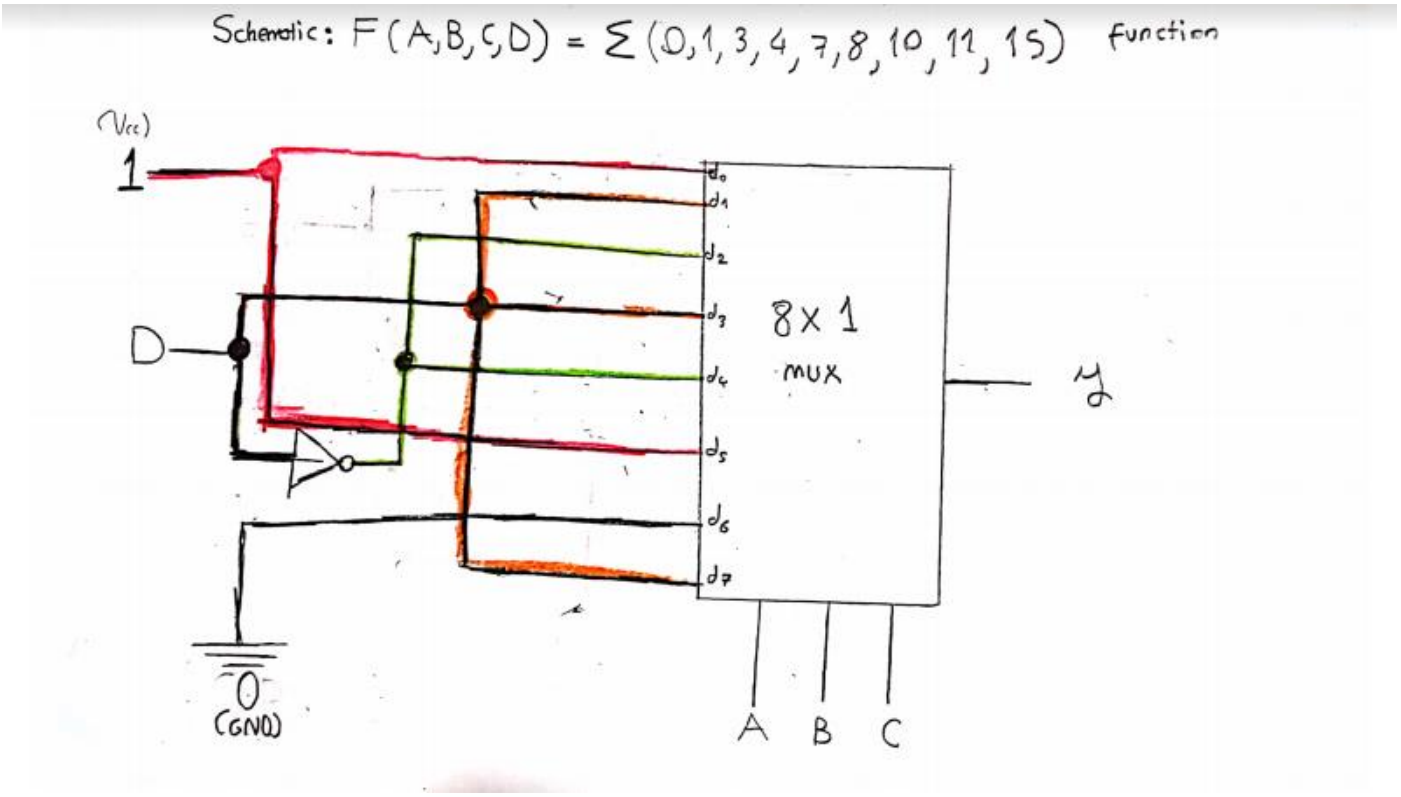



```
module andGate(input logic a, b, output logic product);
  assign product =  a & b;
endmodule

module orGate(input logic a, b, output logic sum);
  assign sum =  a | b;
endmodule

module inverter(input logic a, output logic inputBar);
  assign inputBar = ~a;
endmodule

module eightToOneMultiplexer(
   input logic s2,s1,s0,d0,d1,d2,d3,d4,d5,d6,d7,
   output logic y
```

```
    );

    logic s2Inverted, y1, y2, y1after, y2after;

    inverter inv(s2, s2Inverted);
    fourToOneMultiplexer mux1(s1, s0, d0, d1, d2, d3, y1);
    fourToOneMultiplexer mux2(s1, s0, d4, d5, d6, d7, y2);
    andGate aG1(s2Inverted, y1, y1after);
    andGate aG2(s2, y2, y2after);
    orGate oG(y1after, y2after, y);

endmodule
// TESTBENCH FOR 8-TO-1 MUX
module eightToOneMultiplexer_testbench();
  logic s2,s1,s0,d0,d1,d2,d3,d4,d5,d6,d7;
  logic y;

  // instantiate device under test
  eightToOneMultiplexer dut(s2,s1,s0,d0,d1,d2,d3,d4,d5,d6,d7,y);

  initial begin

  //d0 check
  s2 = 0; s1 = 0; s0 = 0; d0 = 0; d1 = 1'bx; d2 = 1'bx; d3 = 1'bx;
  d4 = 1'bx; d5 = 1'bx; d6 = 1'bx; d7= 1'bx; #10;
  d0 = 1; #10;

  //d1 check
  s0 = 1; d0 = 1'bx; d1 = 0; #10;
  d1 = 1; #10;
```

```
  //d2 check
  s0 = 0; s1 = 1; d1 = 1'bx; d2 = 0; #10;
  d2 = 1; #10;

  //d3 check
  s0 = 1; d2 = 1'bx; d3 = 0; #10;
  d3 = 1; #10;

  //d4 check
  s0 = 0; s1 = 0; s2 = 1; d3 = 1'bx; d4 = 0; #10;
  d4 = 1; #10;

  //d5 check
  s0 = 1;  d4 = 1'bx; d5 = 0; #10;
  d5 = 1; #10;

  //d6 check
  s0 = 0; s1 = 1; d5 = 1'bx; d6 = 0; #10;
  d6 = 1; #10;

  //d7 check
  s0 = 1; d6 = 1'bx; d7 = 0; #10;
  d7 = 1; #10;

   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 booleanFunctionImplementation(
   input logic a, b, c, d,
   output logic y
   );
   logic dInverted;

   inverter inv(d, dInverted);
   eightToOneMultiplexer mux8toOne( a, b, c, 1, d, dInverted, d, dInverted, 1, 0, d, y);
endmodule
```