19.10.2020

CS223-05 LAB 03

Digital Design and Computer Architecture

Remzi Tepe - 21802713

Behavioral SystemVerilog Module for 2-to-4 decoder:

```
module twoTofourdecoder_behavioral(input logic in1, in0,

                                   output logic y3, y2, y1, y0);

  assign y3 = in1 & in0;

  assign y2 = in1 & ~in0;

  assign y1 = ~in1 & in0;

 assign y0 = ~in1 & ~in0;

endmodule
```

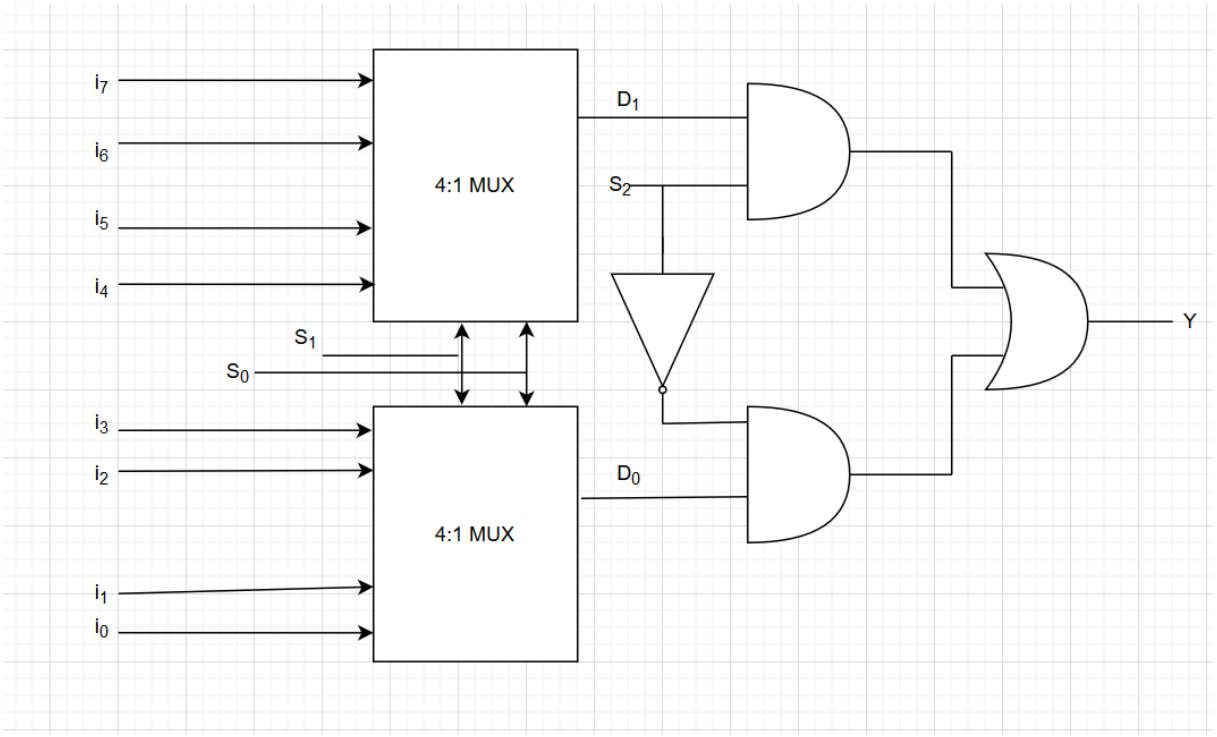
Testbench for 2-to-4 decoder:

```
module testbench_decoderBehavioral();

        logic in1, in0;

       logic y3, y2, y1, y0;

      twoTofourdecoder_behavioral dut(in1, in0, y3, y2, y1, y0);


       initial begin

       in1 = 0; in0 = 0; #100;

      in0 = 1; #100;

     in1 = 1; in0 = 0; #100

     in1 = 1; in0 = 1; #100;


    end

endmodule
```

Behavioral SystemVerilog Module for 4:1 Muliplexer:

```
module muxFourtoOne(input logic d0, d1, d2, d3,
                                input logic s0, s1,
                                output logic y);
  assign y = s1 ? (s0 ? d3 : d2) : (s0 ? d1 : d0);
endmodule
```

Block Diagram of 8-to-1 Multiplexer by using two 4-to-1 MUX, two AND gates, an inverter, and an OR gate.



SystemVerilog Module of it:

```
module inv(input logic a,
           output logic b);
          assign b = ~a;
endmodule

module and2(input logic a, b,
           output logic c);
          assign c = a & b;
endmodule
```

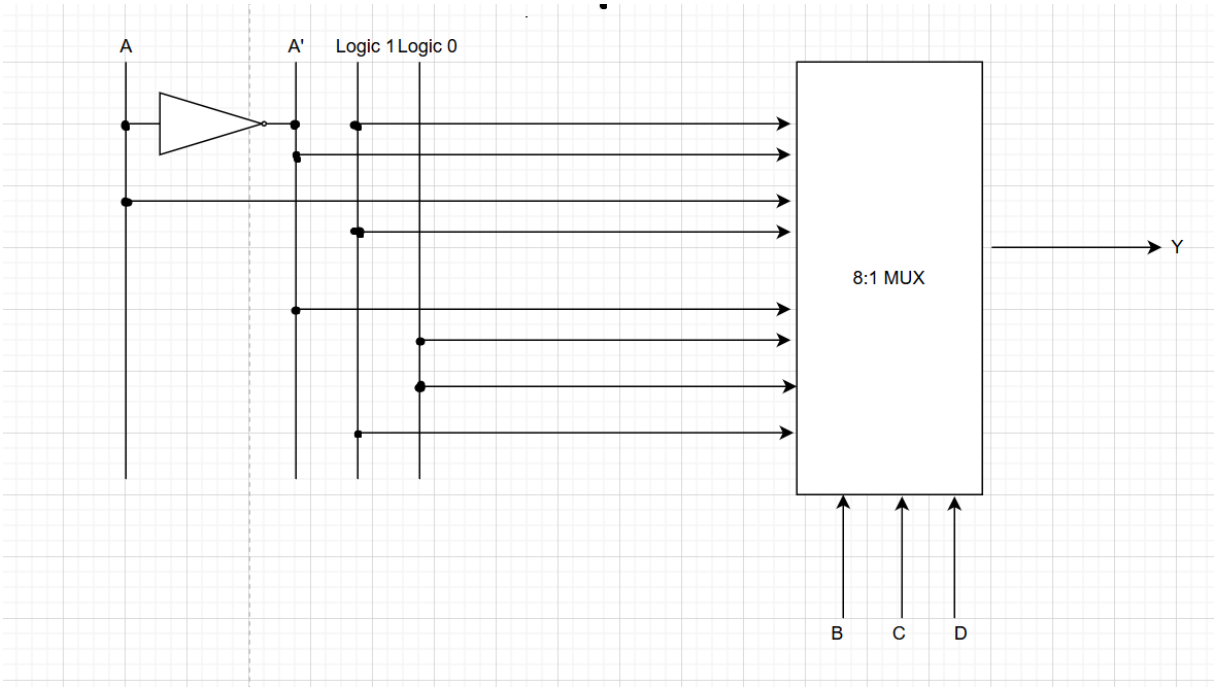
```
module or2(input logic a, b,
           output logic c);
    assign c = a | b;
endmodule

module muxEighttoOne(input logic i0, i1, i2, i3, i4, i5, i6, i7,
                     input logic s2, s1, s0,
                     output logic y);
    logic n1, n2, n3, n4, n5;
    muxFourtoOne multiplexer4to1_1(i0, i1, i2, i3, s0, s1, n1);
    muxFourtoOne multiplexer4to1_2(i4, i5, i6, i7, s0, s1, n2);
    and2 andgate_1(n2, s2, n3);
    inv inverter(s2, n4);
    and2 andgate_2(n4, n1, n5);
    or2 orgate(n3, n5, y);
endmodule
```

Testbench for this structure:

```
module testBenchforMuxEighttoOne();

 logic i0, i1, i2, i3, i4, i5, i6, i7;

        logic s2, s1, s0;

        logic y;

        muxEighttoOne name(.i0(i0), .i1(i1), .i2(i2), .i3(i3), .i4(i4), .i5(i5), .i6(i6), .i7(i7), .s2(s2), .s1(s1),
.s0(s0), .y(y));

        initial begin

         i0 = 0; i1 = 0; i2 = 0; i3 = 0; i4 = 0; i5 = 0; i6 = 0; i7 = 0; s0 = 0; s1 = 0; s2 = 0;

         #2000 $finish;

         end

         always #1 i7 = ~i7;

         always #2 i6 = ~i6;

         always #4 i5 = ~i5;

         always #8 i4 = ~i4;

         always #16 i3 = ~i3;

         always #32 i2 = ~i2;

         always #64 i1 = ~i1;
```

```
        always #128 i0 = ~i0;

        always #256 s0 = ~s0;

        always #512 s1 = ~s1;

        always #1024 s2 = ~s2;

endmodule
```

Block Diagram of the function given in the Part E:



SystemVerilog Module for this function:

```
module muxFunction(input logic a, b, c, d,
                   output logic y);
                   logic n1;
                   inv inverter_mux(a, n1);
          muxEighttoOne multiplexer8to1(1, n1, a, 1, n1, 0, 0, 1, b, c, d, y);
endmodule
```