CS223 DIGITAL DESIGN

SECTION 02

LAB 03

Sümeyye ACAR

22103640

07.05.2023

1.1 Behavioral 1-to-2 Decoder Module (with enable)

```
module decoder_with_enable_1to2(input logic in, en, output logic [1:0] out);
always_comb begin
   if(en) begin
      case(in)
         1'b0: out = 2'b10;
         1'b1: out = 2'b01;
         default: out = 2'b00;
      endcase
   end
   else begin
      out = 2'b00;
   end
end
endmodule
```

1.2 Testbench

```
module onetwo_testbench();
logic in, en;
logic [1:0] out;
decoder_with_enable_1to2 u0(.in(in), .en(en), .out(out));

initial
   begin
      in = 0; en = 0;
            #10
      in = 1;
            #10
      in = 0; en = 1;
            #10
      in = 1;
            #10
      $finish;
   end
endmodule
```

2.1 Structural 2-to-4 Decoder Module (with enable signal) Using Three 1-to-2 Decoders

```
module decoder_with_enable_2to4(input logic [1:0] in,
                                input logic en,
                                output logic [3:0] out);
wire [1:0] temp1;
decoder_with_enable_1to2 u0(.in(in[1]), .en(en),       .out(temp1)   );
decoder_with_enable_1to2 u1(.in(in[0]), .en(temp1[0]), .out(out[1:0]));
decoder_with_enable_1to2 u2(.in(in[0]), .en(temp1[1]), .out(out[3:2]));
endmodule
```

2.2Testbench

```
module twofour_testbench();
logic en;
logic [1:0] in;
logic [3:0] out;
decoder_with_enable_2to4 u0(.in(in), .en(en), .out(out));
initial begin
      en = 0; in = 2'b00;
                #10
            in = 2'b01;
                #10
            in = 2'b10;
                #10
            in = 2'b11;
                #10
      en = 1; in = 2'b00;
                #10
            in = 2'b01;
                #10
            in = 2'b10;
                #10
            in = 2'b11;
                #10
      $finish;
   end
endmodule
```

3.1 Behavioral 2-to-1 Multiplexer Module

```
module multiplexer_2to1(input logic s, in, in2, output logic out);
always_comb begin
   if(s == 1'b0) begin
      out = in;
   end
   else begin
      out = in2;
   end
end
endmodule
```

3.2Testbench (additional)

```
module multiplexer_twoone_testbench();
logic s, in, in2, out;
multiplexer_2to1 u0(.s(s), .in(in), .in2(in2), .out(out));
initial begin
      s = 0; in = 0; in2 = 0;
                  #10
               in2 = 1;
                  #10
         in = 1; in2 = 0;
                  #10
      s = 1; in = 0;
                  #10
      s = 1; in = 1; in2 = 1;
                  #10
               in2 = 0;
                  #10
         in = 0; in2 = 1;
                  #10
      s = 0; in = 1;
                  #10
      $finish;
   end
endmodule
```

4.1 Structural 4-to-1 Multiplexer Module Using Three 2-to-1 Multiplexer

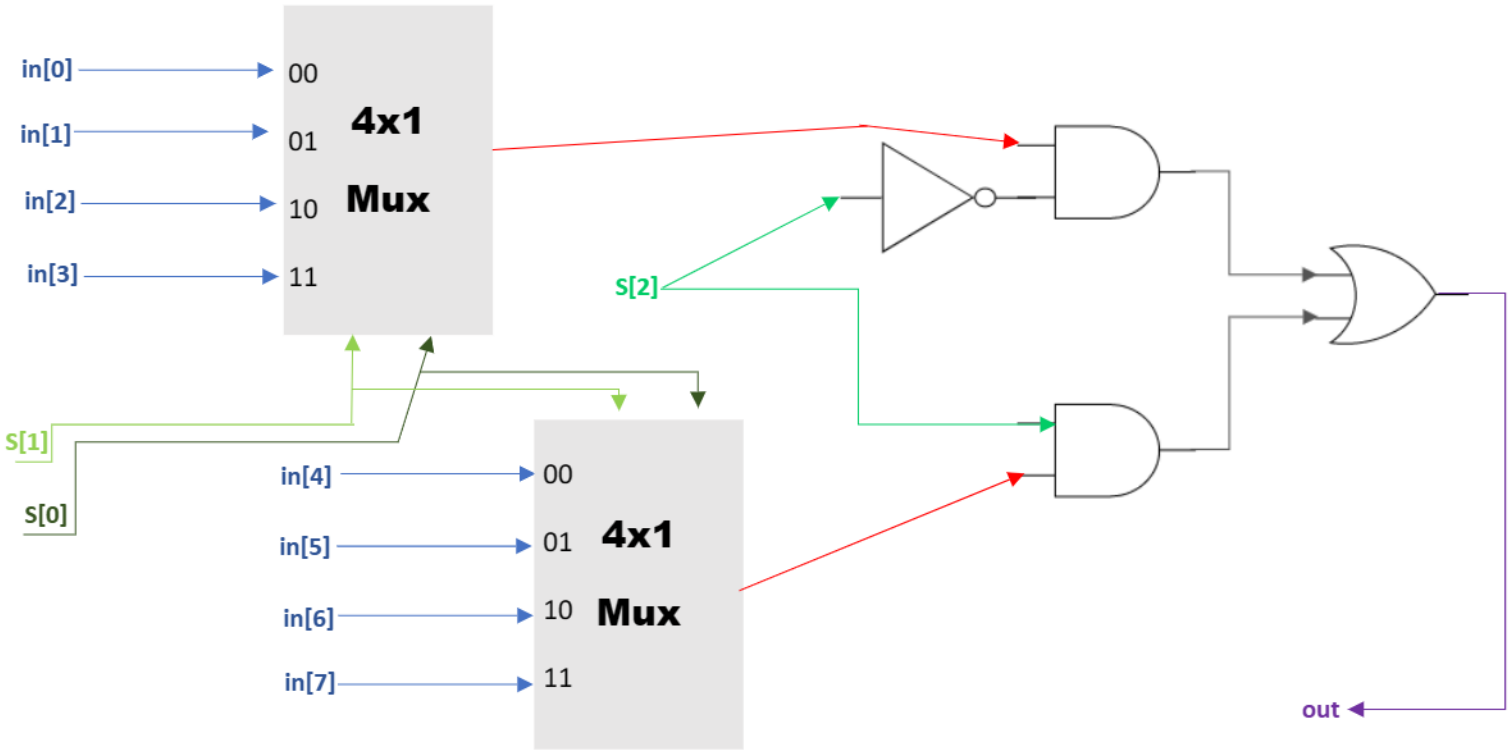
```
module multiplexer_4to1(input logic [1:0] s, input logic [3:0] in,
                        output logic out);
wire temp1, temp2;
multiplexer_2to1 u0(.s(s[0]), .in(in[0]), .in2(in[1]), .out(temp1));
multiplexer_2to1 u1(.s(s[0]), .in(in[2]), .in2(in[3]), .out(temp2));
multiplexer_2to1 u2(.s(s[1]), .in(temp1), .in2(temp2), .out(out));
endmodule
```

4.2 Testbench

```
module multiplexer_fourone_testbench();
logic [1:0] s;
logic [3:0] in;
multiplexer_4to1 u0(.s(s),.in(in), .out(out));

initial
   begin
      s = 2'b00; in = 4'b0001;
                  #10
      s = 2'b01; in = 4'b0010;
                  #10
      s = 2'b10; in = 4'b0100;
                  #10
      s = 2'b11; in = 4'b1000;
                  #10
      $finish;
   end
endmodule
```

5.1 Block diagram of Structural 8-to1 MUX Module Using Two 4-to-1 MUX modules, Two AND Gates and an OR Gate



5.2 Structural 8-to1 MUX Module Using Two 4-to-1 MUX modules, Two AND Gates and an OR Gate

```
module multiplexer_8to1(input logic [2:0] s,
                        input logic [7:0] in,
                        output logic out);

wire temp1, temp2;
wire out1, out2;

multiplexer_4to1 u0(.s(s[1:0]), .in(in[3:0]), .out(temp1));
multiplexer_4to1 u1(.s(s[1:0]), .in(in[7:4]), .out(temp2));
assign out1 = temp1 & ~s[2];
assign out2 = temp2 & s[2];
out = out1 | out2;
endmodule
```

5.3Tesstbench

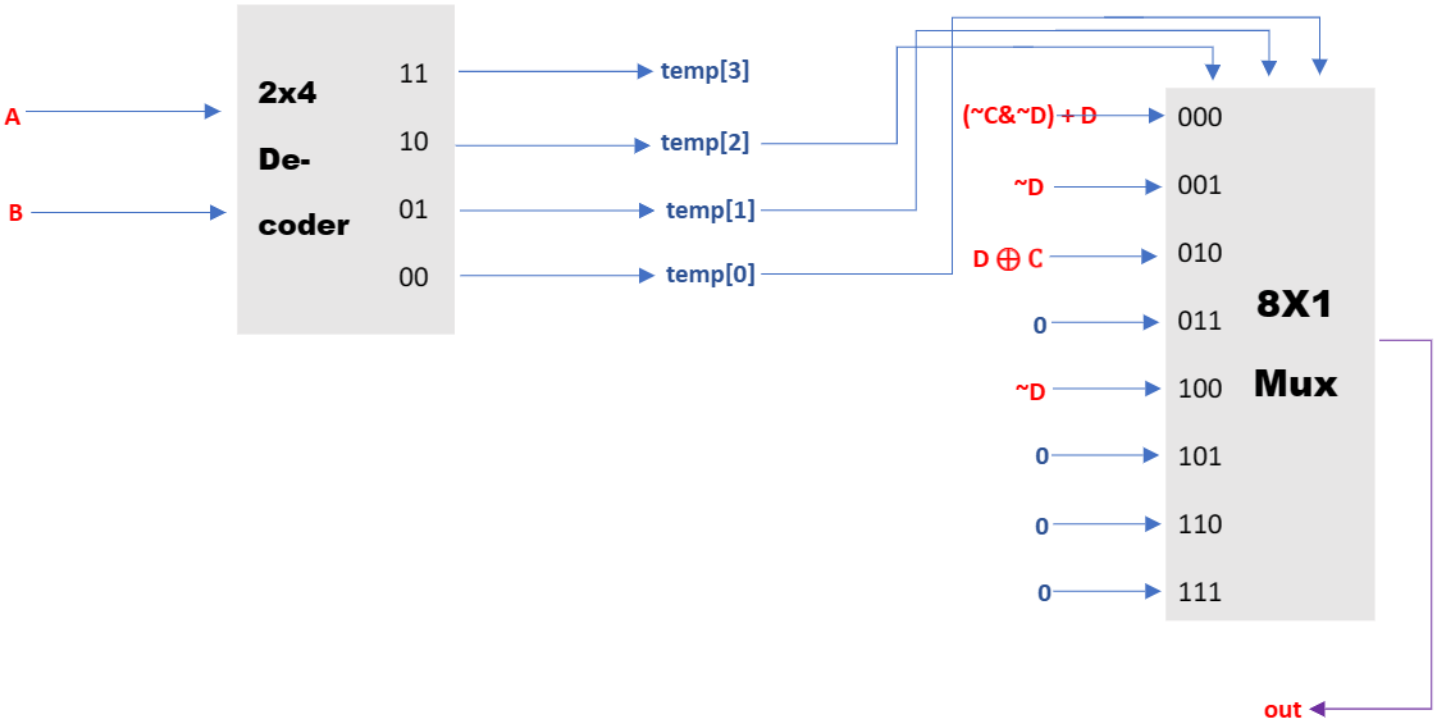
```
module multiplexer_eightone_testbench();

logic [2:0] s;
logic [7:0] in;
logic out;
multiplexer_8to1 u0(.s(s), .in(in), .out(out));

initial
   begin
      s = 3'b000; in = 8'b00000001;
                  #10
      s = 3'b001; in = 8'b00000010;
                  #10
      s = 3'b010; in = 8'b00000100;
                  #10
      s = 3'b011; in = 8'b00001000;
                  #10
      s = 3'b100; in = 8'b00010000;
                  #10
      s = 3'b101; in = 8'b00100000;
                  #10
      s = 3'b110; in = 8'b01000000;
                  #10
      s = 3'b111; in = 8'b10000000;
                  #10
      $finish;
   end

endmodule
```

6.1 Block diagram of F(A,B,C,D)=∑(0,2,5,6,8,10,12,13,15) Function Using Only One 8-to-1 Multiplexer and a 2-to-4 Decoder



6.2 SystemVerilog Module of F(A,B,C,D)=∑(0,2,5,6,8,10,12,13,15) Function Using Only One 8-to-1 Multiplexer and a 2-to-4 Decoder

```
module method(input logic a, b, c, d, output logic out);
wire [3:0] temp_out;
wire [7:0] temp_in;
assign temp_in[0] = ~d;
assign temp_in[1] = ~c&~d + d;
assign temp_in[2] = ~d;
assign temp_in[3] = 1'b0;
assign temp_in[4] = c^d;
assign temp_in[5] = 1'b0;
assign temp_in[6] = 1'b0;
assign temp_in[7] = 1'b0;

decoder_with_enable_2to4 u0(.in({a,b}), .en(1'b1), .out(temp_out));
multiplexer_8to1 u1(.s(temp_out[2:0]),.in(temp_in), .out(out));
endmodule
```

6.3 Testbench

```
module method_testbench();
logic a, b, c, d, out;
method u0(.a(a), .b(b), .c(c), .d(d), .out(out));
initial begin
    a = 0; b = 0; c = 0; d = 0;                    #10
                          d = 1;                    #10
                  c = 1; d = 0;                    #10
                          d = 1;                    #10
        b = 1; c = 0; d = 0;                       #10
                  d = 1;
                    #10
              c = 1; d = 0;
                    #10
                  d = 1;
                    #10
    a = 1; b = 0; c = 0; d = 0;
                    #10
                  d = 1;
                    #10
              c = 1; d = 0;
                    #10
                  d = 1;
                    #10
        b = 1; c = 0; d = 0;
                    #10
                  d = 1;
                    #10
              c = 1; d = 0;
                    #10
                  d = 1;
                    #10
    $finish;
  end
endmodule
```