11.03.2020

CS 223-4

LAB-3

PRELIMINARY REPORT

ZÜBEYİR BODUR

21702382

Spring 2020

Zübeyir Bodur

1. 3:8 Decoder

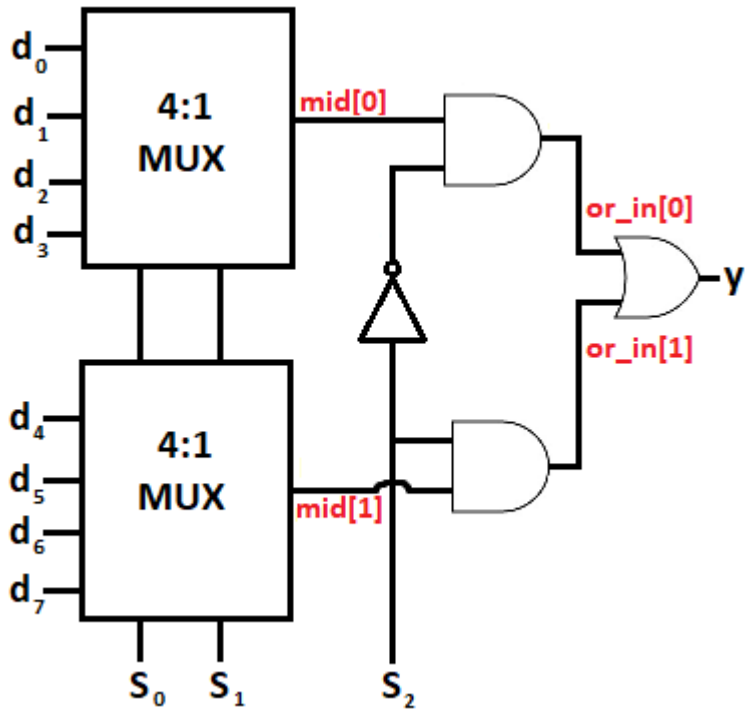
```
// Behavorial module for 3 to 8 decoder
module threeTo8decoder( input logic [2:0]a,
                        output logic [7:0]y
  );
  assign y[0] = ~a[2] & ~a[1] & ~a[0];
  assign y[1] = ~a[2] & ~a[1] & a[0];
  assign y[2] = ~a[2] & a[1]  & ~a[0];
  assign y[3] = ~a[2] & a[1]  & a[0];
  assign y[4] = a[2] & ~a[1] & ~a[0];
  assign y[5] = a[2] & ~a[1] & a[0];
  assign y[6] = a[2] & a[1]  & ~a[0];
  assign y[7] = a[2] & a[1]  & a[0];
endmodule

// Testbench for 3:8 decoder
module test3to8decoder();
  logic [2:0]a;
  logic [7:0]y;
  threeTo8decoder test( a, y);
  initial begin
    a = 3'b0; #10;
    repeat(2) begin
      repeat(2) begin
        repeat(2) begin
          a[0] = ~a[0]; #10;
        end
        a[1] = ~a[1]; #10;
      end
      a[2] = ~a[2]; #10;
    end
  $stop;
  end
endmodule
```

2. 4:1 MUX

```
// Behavorial module for 4 to 1 mux
module fourTo1mux( input logic [1:0]s, [3:0]d,
            output logic y
  );
  logic [1:0]mid;
  assign mid[0] = s[0] ? d[1] : d[0];
```

```
   assign mid[1] = s[0] ? d[3] : d[2];
   assign y = s[1] ? mid[1] : mid[0];
endmodule
```

3. 8:1 MUX



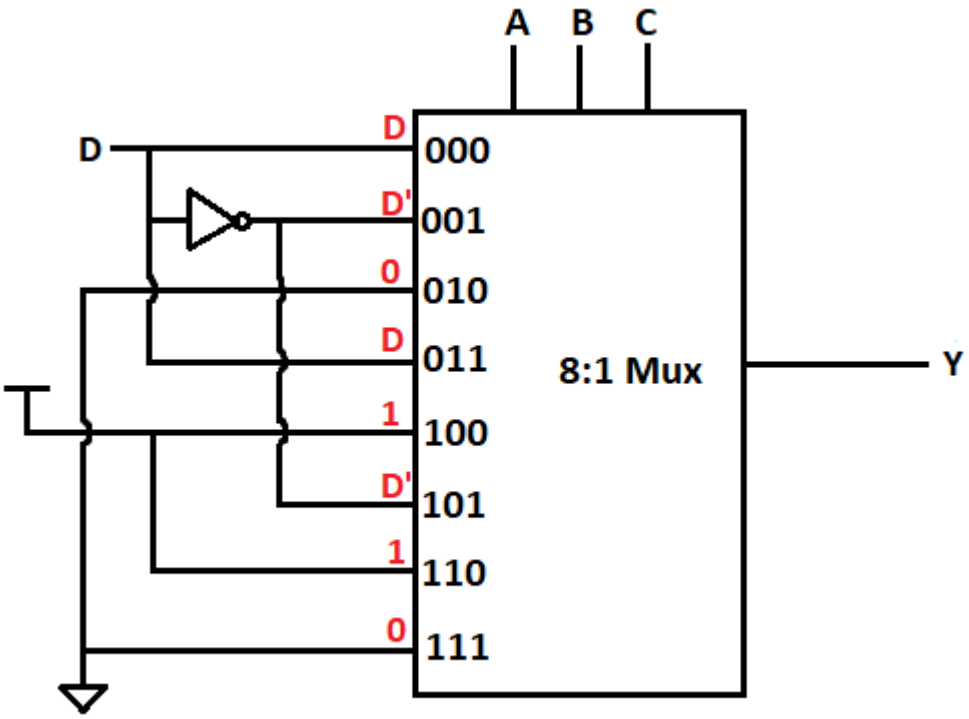

```
// Structural module for 8 to 1 mux
module eightTo1mux( input logic [2:0]s, [7:0]d,
                                output logic y
   );
   logic [1:0]mid;
   logic [1:0]or_in;
   fourTo1mux mux4_0( s[1:0], d[3:0], mid[0]);
   fourTo1mux mux4_1( s[1:0], d[7:4], mid[1]);
   and and_0 ( or_in[0], mid[0], ~s[2]);
   and and_1 ( or_in[1], mid[1], s[2]);
   or or_0( y, or_in[0], or_in[1]);
endmodule
```

Zübeyir Bodur

```
// Testbench for 8:1 MUX
module test8to1mux();
  logic [2:0]s;
  logic [7:0]d;
  logic y;
  eightTo1mux test( s, d, y);
  initial begin
    s = 3'b0; d = 8'b0; #10;
    repeat(2) begin
      repeat(2) begin
        repeat(2) begin
          repeat(2) begin
            repeat(2) begin
              repeat(2) begin
                repeat(2) begin
                  repeat(2) begin
                    repeat(2) begin
                      repeat(2) begin
                        repeat(2) begin
                          repeat(2) begin
                            d[0] = ~d[0]; #1;
                          end
                          d[1] = ~d[1]; #1;
                        end
                        d[2] = ~d[2]; #1;
                      end
                      d[3] = ~d[3]; #1;
                    end
                    d[4] = ~d[4]; #1;
                  end
                  d[5] = ~d[5]; #1;
                end
                d[6] = ~d[6]; #1;
              end
              d[7] = ~d[7]; #1;
            end
            s[0] = ~s[0]; #1;
          end
          s[1] = ~s[1]; #1;
        end
        s[2] = ~s[2]; #1;
```

```
    end
  $stop;
  end
endmodule
```

4. Function F(A,B,C,D)




```
// Structural module for the boolean function F(A,B,C,D)
// = ∑(1, 2, 7, 8, 9, 10, 12, 13)
module functionF( input logic a, b, c, d,
                  output logic y
  );
  logic [2:0] s;
  logic [7:0] d_prime;
  assign s[2] = a;
  assign s[1] = b;
  assign s[0] = c;
  assign d_prime[7] = 0;
  assign d_prime[6] = 1;
  assign d_prime[5] = ~d;
  assign d_prime[4] = 1;
```

```
   assign d_prime[3] = d;
   assign d_prime[2] = 0;
   assign d_prime[1] = ~d;
   assign d_prime[0] = d;
   eightTo1mux mux8_0( s, d_prime, y );
endmodule

// Testbench for boolean function F(A,B,C,D)
// = ∑(1, 2, 7, 8, 9, 10, 12, 13)
module testfunctionF();
   logic a, b, c, d, y;
   functionF test(a, b, c, d, y);
   initial begin
      a = 0; b = 0; c = 0; d = 0; #10;
      repeat(2) begin
         repeat(2) begin
            repeat(2) begin
               repeat(2) begin
                  a = ~a; #10;
               end
               b = ~b; #10;
            end
            c = ~c; #10;
         end
         d = ~d; #10;
      end
   $stop;
   end
endmodule
```