CS 223 Digital Design

Bilkent University

Fall 2024-25

Laboratory Assignment 3

Digital Building Blocks: Decoders and MUXs in SystemVerilog

Nabeeha Khan

22301304

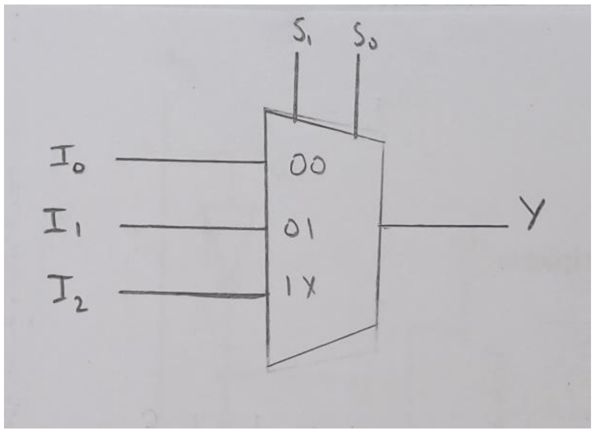
Section: 03

Date: 18th November 2024

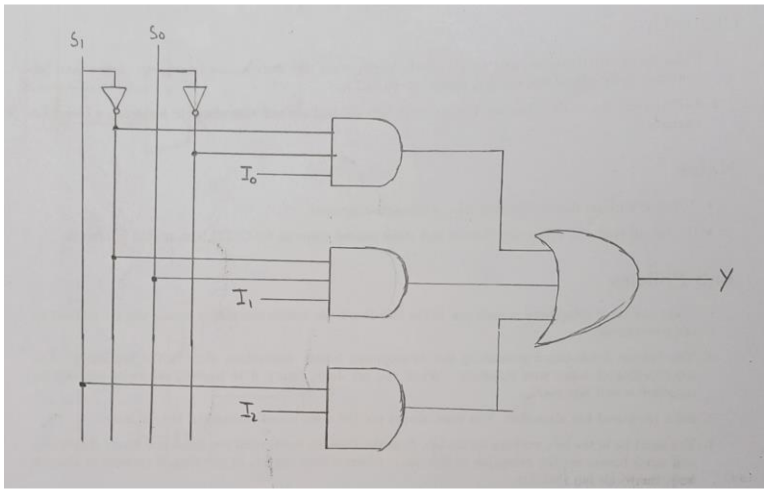
Multiplexer:

1) 3-to-1 Multiplexer:

Graphical Symbol:



Logic Diagram:



Truth Table:

S_1	**S_0**	**Y**
0	0	I_0
0	1	I_1
1	X	I_2

SystemVerilog Module:

```
module mux3to1(input logic I0, I1, I2, S0, S1, output logic Y);
```

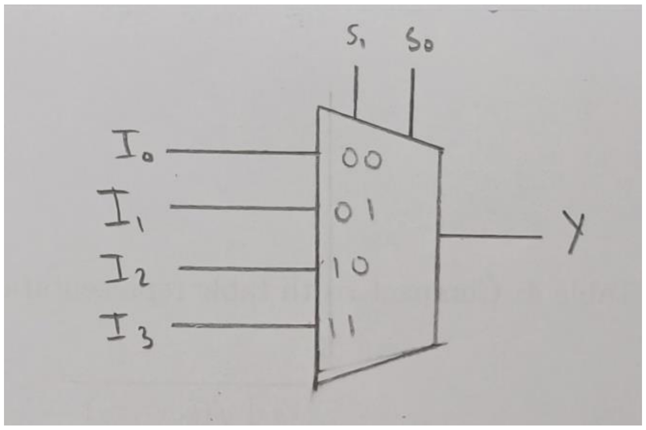
```
             logic l0, l1, l2;
            assign l0 = ~S1 & ~S0 & I0;
            assign l1 = ~S1 & S0 & I1;
            assign l2 = S1 & I2;
            assign Y = l0 | l1 | l2;
endmodule
```

Testbench:

```
module mux3to1_TB();
logic I0, I1, I2, S0, S1, Y;
mux3to1 dut(I0, I1, I2, S0, S1, Y);
initial begin
I0 = 0; I1 = 0; I2 = 0; S0 = 0; S1 = 0; #10;
I0 = 1; I1 = 0; I2 = 0; S0 = 0; S1 = 0; #10;
I0 = 0; I1 = 0; I2 = 0; S0 = 0; S1 = 1; #10;
I0 = 0; I1 = 1; I2 = 0; S0 = 0; S1 = 1; #10;
I0 = 0; I1 = 0; I2 = 0; S0 = 1; S1 = 0; #10;
I0 = 0; I1 = 0; I2 = 1; S0 = 1; S1 = 0; #10;
I0 = 0; I1 = 0; I2 = 0; S0 = 1; S1 = 1; #10;
I0 = 0; I1 = 0; I2 = 1; S0 = 1; S1 = 1; #10;
end
endmodule
```

2) 4-to-1 Multiplexer:

Graphical Symbol:



Truth Table:

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

SystemVerilog Module:

```
module mux4to1(input logic I0, I1, I2, I3, S0, S1, output logic Y);
logic l0, l1;
assign l0 = (~S0 & I0) | (S0 & I1);
assign l1 = (~S0 & I2) | (S0 & I3);
assign Y = (~S1 & l0) | (S1 & l1);
endmodule
```

Testbench:

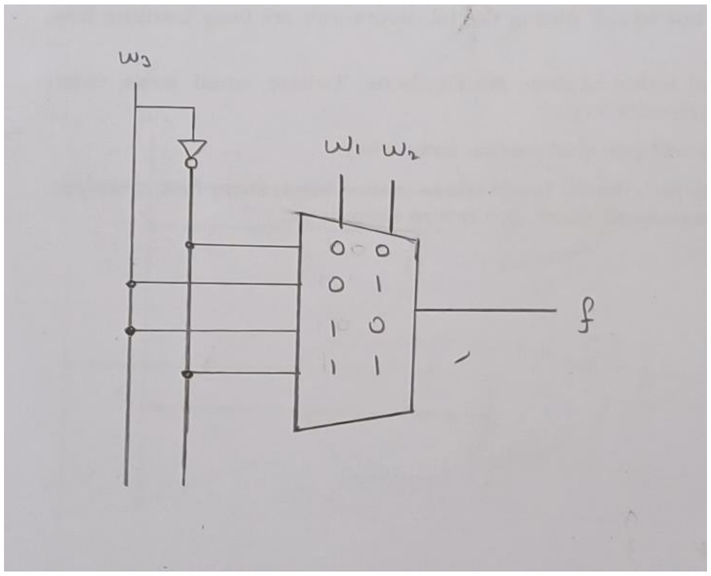
```
module mux4to1_TB();
logic I0, I1, I2, I3, S0, S1, Y;
mux4to1 dut(I0, I1, I2, I3, S0, S1, Y);
initial begin
I0 = 0; I1 = 0; I2 = 0; I3 = 0; S0 = 0; S1 = 0; #10;
I0 = 1; I1 = 0; I2 = 0; I3 = 0; S0 = 0; S1 = 0; #10;
```

```
I0 = 0; I1 = 0; I2 = 0; I3 = 0; S0 = 1; S1 = 0; #10;
I0 = 0; I1 = 1; I2 = 0; I3 = 0; S0 = 1; S1 = 0; #10;
I0 = 0; I1 = 0; I2 = 0; I3 = 0; S0 = 0; S1 = 1; #10;
I0 = 0; I1 = 0; I2 = 1; I3 = 0; S0 = 0; S1 = 1; #10;
I0 = 0; I1 = 0; I2 = 0; I3 = 0; S0 = 1; S1 = 1; #10;
I0 = 0; I1 = 0; I2 = 0; I3 = 1; S0 = 1; S1 = 1; #10;
        end
endmodule
```

3) Implementation of f function:

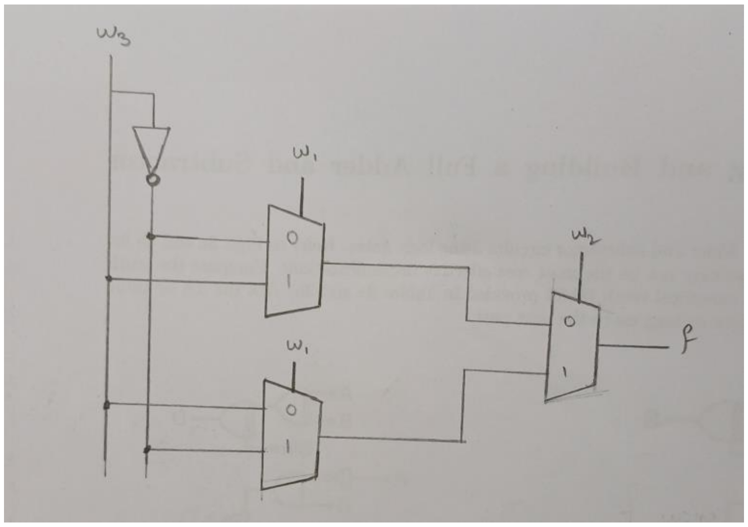
Using 4-to-1 Mux:

We would require only one 4-to-1 mux and one inverter for this. We provide w_1 and w_2 as select inputs for the mux and pass w_3 and its complement as the inputs to the mux so that we get ~w_3 for 00 and 11, and w_3 for 01 and 10, and we get f as the output to this mux. The logic diagram will be as follows:



Using 2-to-1 Mux:

We would require three 2-to-1 mux and one inverter for this. We provide w1 as the select input for the first two mux. The first mux will get ~w3 for 0 and w3 for 1 while the second mux will get w3 for 0 and ~w3 for 1 as input. The result of both mux will go to a third mux, for which we provide w2 as select input, and it will pass the input of first mux for 0 and input of second mux for 1. The output of this final mux will yield function f. The logic diagram is as follows:



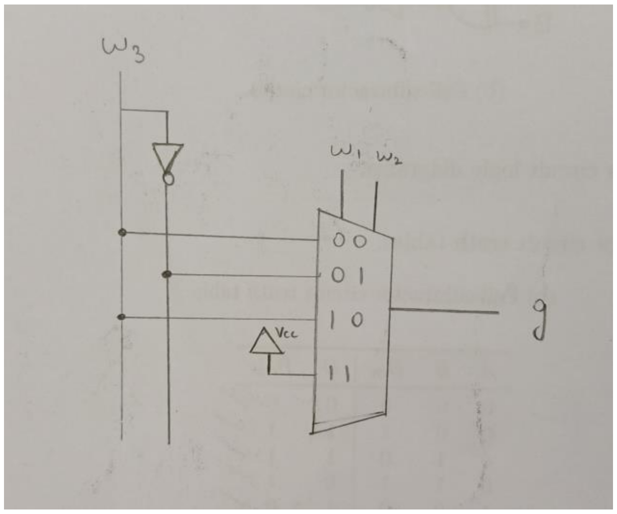
Operation of function f:

Function f is based on the three input XNOR gate functionality i.e. it gives 1 when even number of inputs are 1 and 0 othwerwise.

4) Implementation of g function:

Using 4-to-1 mux:

We would require a 4-to-1 mux and an inverter for this. We would provide w_1 and w_2 as the select input to the mux, and give w_3 for 00 and 10 cases, $\sim w_3$ for 01 case and V_{cc} (1) for 11 case. The output of the mux would yield function g. The logic diagram is as follows:



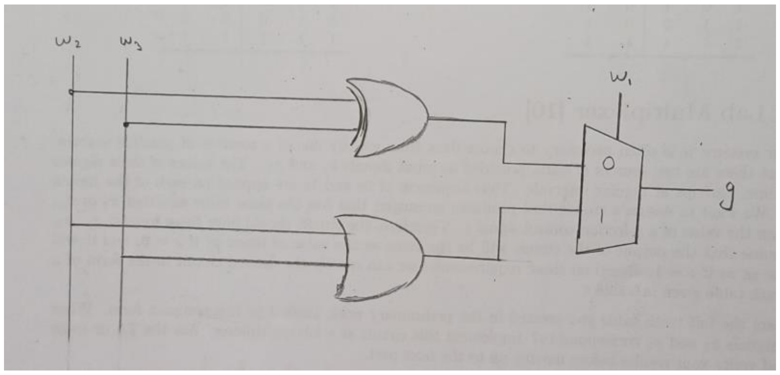
Modified truth table:

w_1	**w_2**	**g**
0	0	w_3

0	1	$\sim w_3$
1	0	w_3
1	1	1

Using 2-to-1 mux and gates:

We can implement this function using a 2-to-1 mux, one XOR gate and one OR gate. Both XOR and OR gate will receive w_2 and w_3. The mux will have w_1 as control input and will pass output of XOR gate for 0 and output of OR for 1. The output of mux will give function g. The logic diagram is as follows:



Modified truth table:

w_1	g
0	$w_2 \wedge w_3$
1	$w_2 \mid w_3$

SystemVerilog Module:

```
module gFunction(input logic w1, w2, w3, output logic g);

logic l0, l1, l2, l3;

assign l0 = w2 ^ w3;

assign l1 = w2 | w3;

assign l2 = l0 & ~w1;

assign l3 = l1 & w1;

assign g = l2 | l3;

endmodule
```

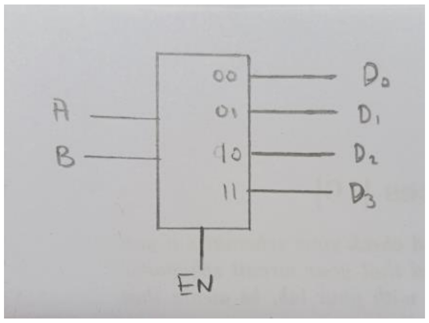
Testbench:

```
module gFunction_TB();

logic w1, w2, w3, g;

gFunction mod(w1, w2, w3, g);

initial begin

w1 = 0; w2 = 0; w3 = 0; #10;
                    w1 = 0; w2 = 0; w3 = 1; #10;

w1 = 0; w2 = 1; w3 = 0; #10;

w1 = 0; w2 = 1; w3 = 1; #10;

w1 = 1; w2 = 0; w3 = 0; #10;

w1 = 1; w2 = 0; w3 = 1; #10;

w1 = 1; w2 = 1; w3 = 0; #10;

w1 = 1; w2 = 1; w3 = 1; #10;

end

endmodule
```

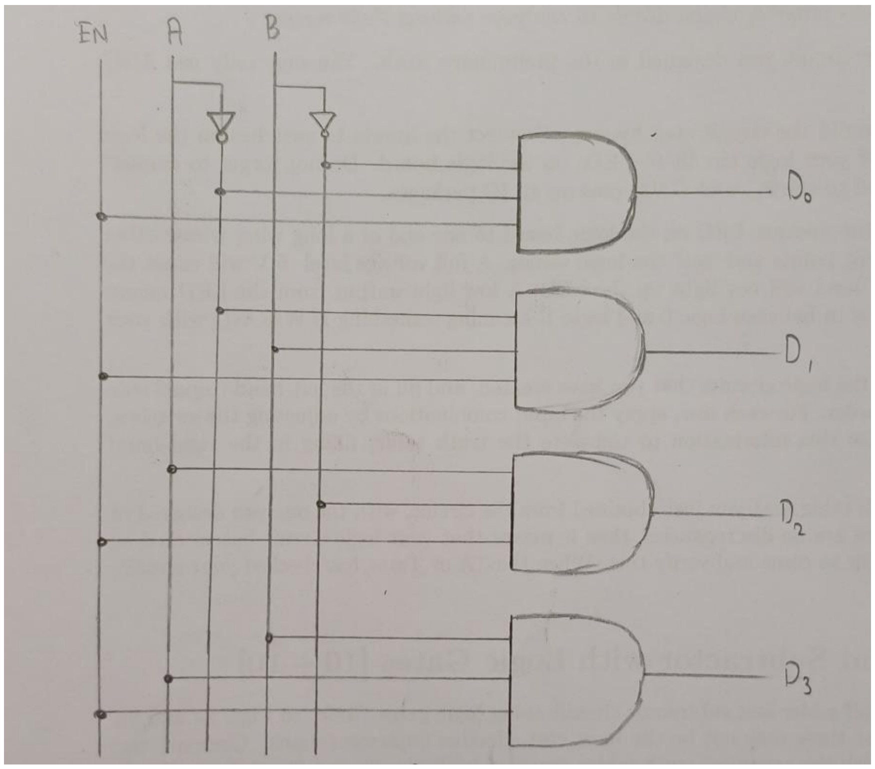
Decoder:

1. 2-to-4 Decoder:

Graphical Symbol:



Logic Diagram:



Truth Table:

A	**B**	**EN**	**D_0**	**D_1**	**D_2**	**D_3**
X	X	0	0	0	0	0
0	0	1	1	0	0	0
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	1

SystemVerilog Module:

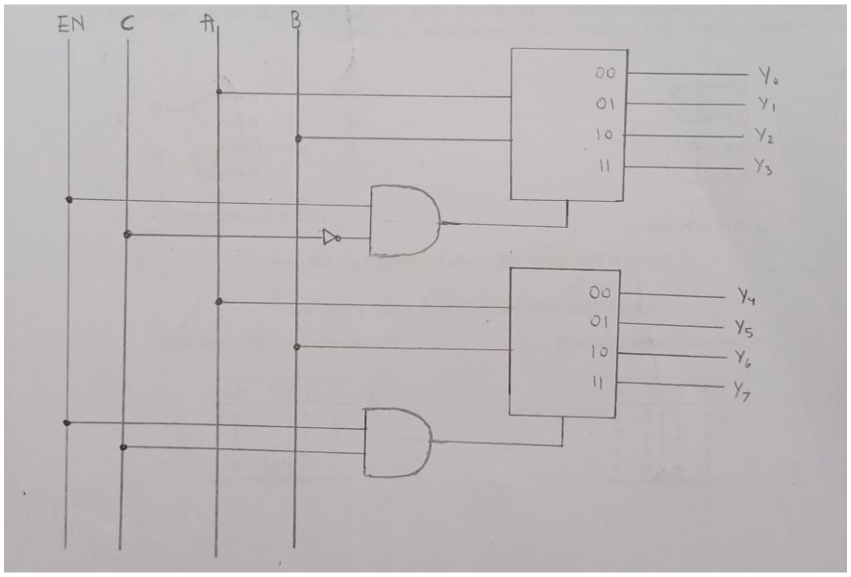
```
module decoder2to4(input logic A, B, EN, output logic D0, D1, D2, D3);

assign D0 = ~A & ~B & EN;

assign D1 = ~A & B & EN;

assign D2 = A & ~B & EN;

assign D3 = A & B & EN;

endmodule
```

Testbench:

```
module decoder2to4_TB();
logic A, B, EN, D0, D1, D2, D3;
decoder2to4 mod(A, B, EN, D0, D1, D2, D3);
initial begin
A = 0; B = 0; EN = 0; #10;
A = 0; B = 1; EN = 0; #10;
A = 1; B = 0; EN = 0; #10;
A = 1; B = 1; EN = 0; #10;
A = 0; B = 0; EN = 1; #10;
A = 0; B = 1; EN = 1; #10;
A = 1; B = 0; EN = 1; #10;
A = 1; B = 1; EN = 1; #10;
end
endmodule
```

2. 3-to-8 Decoder:

Logic Diagram:



SystemVerilog Module:

```
module decoder3to8(input logic A, B, C, EN, output logic Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7);
logic en1, en2;
        assign en1 = EN & ~C;
        decoder2to4 decoder1(A, B, en1, Y0, Y1, Y2, Y3);
        assign en2 = EN & C;
        decoder2to4 decoder2(A, B, en2, Y4, Y5, Y6, Y7);
endmodule
```

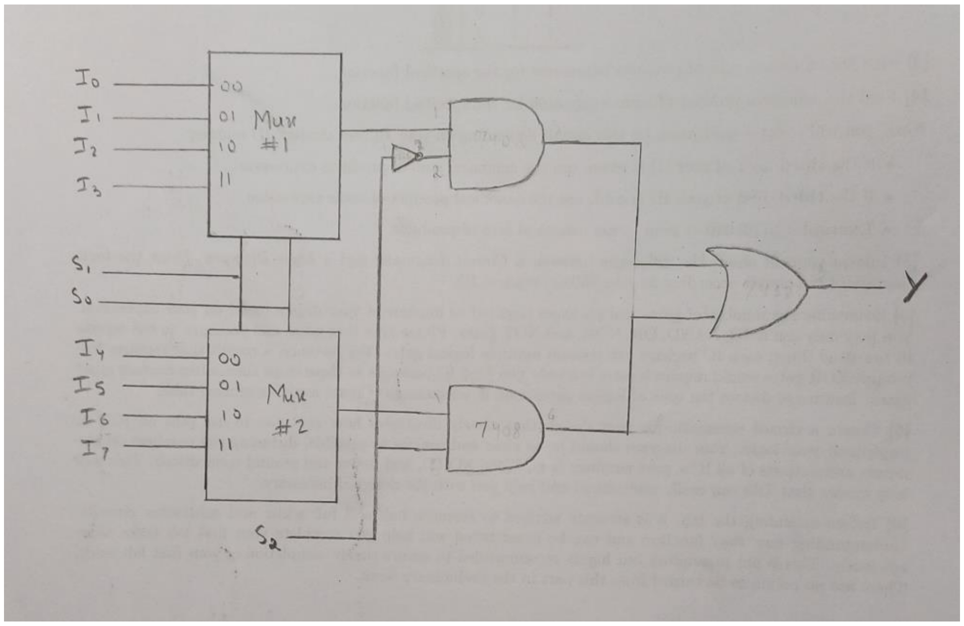
Testbench:

```
module decoder3to8_TB();
logic A, B, C, EN, Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7;
        decoder3to8 mod(A, B, C, EN, Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7);
        initial begin
        A = 0 ; B = 0 ; C = 0 ; EN = 0 ; #10 ;
        A = 1 ; B = 0 ; C = 0 ; EN = 0 ; #10 ;
        A = 0 ; B = 1 ; C = 0 ; EN = 0 ; #10 ;
        A = 1 ; B = 1 ; C = 0 ; EN = 0 ; #10 ;
        A = 0 ; B = 0 ; C = 1 ; EN = 0 ; #10 ;
        A = 1 ; B = 0 ; C = 1 ; EN = 0 ; #10 ;
        A = 0 ; B = 1 ; C = 1 ; EN = 0 ; #10 ;
        A = 1 ; B = 1 ; C = 1 ; EN = 0 ; #10 ;
        A = 0 ; B = 0 ; C = 0 ; EN = 1 ; #10 ;
        A = 1 ; B = 0 ; C = 0 ; EN = 1 ; #10 ;
        A = 0 ; B = 1 ; C = 0 ; EN = 1 ; #10 ;
        A = 1 ; B = 1 ; C = 0 ; EN = 1 ; #10 ;
        A = 0 ; B = 0 ; C = 1 ; EN = 1 ; #10 ;
        A = 1 ; B = 0 ; C = 1 ; EN = 1 ; #10 ;
        A = 0 ; B = 1 ; C = 1 ; EN = 1 ; #10 ;
        A = 1 ; B = 1 ; C = 1 ; EN = 1 ; #10 ;
        end
endmodule
```

Logic Function:

1. 8-to-1 Mux:

Schematic Diagram:



SystemVerilog Module:

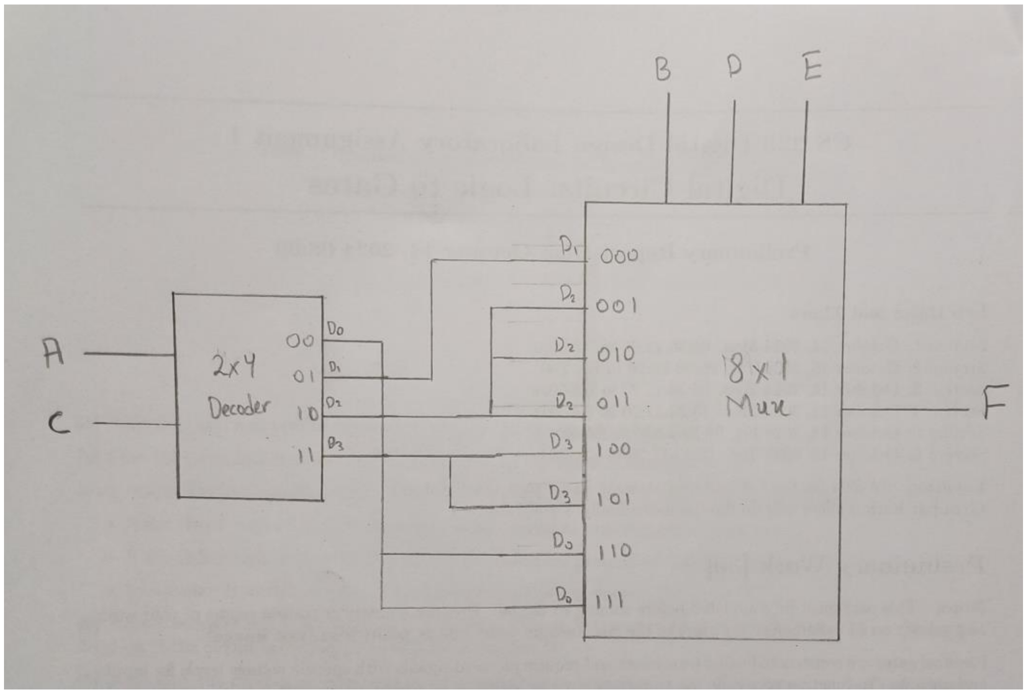
```
module mux8to1(input logic I0, I1, I2, I3, I4, I5, I6, I7, S2, S1, S0, output logic Y);
logic l1, l2, l3, l4;
mux4to1 mux1(I0, I1, I2, I3, S1, S0, l1);
mux4to1 mux2(I4, I5, I6, I7, S1, S0, l2);
assign l3 = l1 & ~S2;
assign l4 = l2 & S2;
assign Y = l3 | l4;
endmodule
```

Testbench:

```
module mux8to1_TB();
logic I0, I1, I2, I3, I4, I5, I6, I7, S2, S1, S0, Y;
mux8to1 mod(I0, I1, I2, I3, I4, I5, I6, I7, S2, S1, S0, Y);
initial begin
        I0 = 0; I1 = 0; I2 = 0; I3 = 0; I4 = 0; I5 = 0; I6 = 0; I7 = 0; S2 =0; S1 = 0; S0 = 0; #10;
        I0 = 1; I1 = 0; I2 = 0; I3 = 0; I4 = 0; I5 = 0; I6 = 0; I7 = 0; S2 =0; S1 = 0; S0 = 0; #10;
        I0 = 0; I1 = 0; I2 = 0; I3 = 0; I4 = 0; I5 = 0; I6 = 0; I7 = 0; S2 =0; S1 = 0; S0 = 1; #10;
        I0 = 0; I1 = 1; I2 = 0; I3 = 0; I4 = 0; I5 = 0; I6 = 0; I7 = 0; S2 =0; S1 = 0; S0 = 1; #10;
        I0 = 0; I1 = 0; I2 = 0; I3 = 0; I4 = 0; I5 = 0; I6 = 0; I7 = 0; S2 =0; S1 = 1; S0 = 0; #10;
        I0 = 0; I1 = 0; I2 = 1; I3 = 0; I4 = 0; I5 = 0; I6 = 0; I7 = 0; S2 =0; S1 = 1; S0 = 0; #10;
        I0 = 0; I1 = 0; I2 = 0; I3 = 0; I4 = 0; I5 = 0; I6 = 0; I7 = 0; S2 =0; S1 = 1; S0 = 1; #10;
        I0 = 0; I1 = 0; I2 = 0; I3 = 1; I4 = 0; I5 = 0; I6 = 0; I7 = 0; S2 =0; S1 = 1; S0 = 1; #10;
        I0 = 0; I1 = 0; I2 = 0; I3 = 0; I4 = 0; I5 = 0; I6 = 0; I7 = 0; S2 =1; S1 = 0; S0 = 0; #10;
        I0 = 0; I1 = 0; I2 = 0; I3 = 0; I4 = 1; I5 = 0; I6 = 0; I7 = 0; S2 =1; S1 = 0; S0 = 0; #10;
        I0 = 0; I1 = 0; I2 = 0; I3 = 0; I4 = 0; I5 = 0; I6 = 0; I7 = 0; S2 =1; S1 = 0; S0 = 1; #10;
        I0 = 0; I1 = 0; I2 = 0; I3 = 0; I4 = 0; I5 = 1; I6 = 0; I7 = 0; S2 =1; S1 = 0; S0 = 1; #10;
        I0 = 0; I1 = 0; I2 = 0; I3 = 0; I4 = 0; I5 = 0; I6 = 0; I7 = 0; S2 =1; S1 = 1; S0 = 0; #10;
        I0 = 0; I1 = 0; I2 = 0; I3 = 0; I4 = 0; I5 = 0; I6 = 1; I7 = 0; S2 =1; S1 = 1; S0 = 0; #10;
        I0 = 0; I1 = 0; I2 = 0; I3 = 0; I4 = 0; I5 = 0; I6 = 0; I7 = 0; S2 =1; S1 = 1; S0 = 1; #10;
        I0 = 0; I1 = 0; I2 = 0; I3 = 0; I4 = 0; I5 = 0; I6 = 0; I7 = 1; S2 =1; S1 = 1; S0 = 1; #10;
        end
endmodule
```

2. Five Input Function:

Schematic Diagram:



SystemVerilog Module:

```
module Ffunction(input logic A, B, C, D, E, output logic F);
logic EN, D0, D1, D2, D3;
assign EN = 1’b1;
decoder2to4 decoder(A, C, EN, D0, D1, D2, D3);
mux8to1 mux( D1, D2, D2, D2, D3, D3, D0, D0, B, D, E, F);
endmodule
```