Verilog Assignment 1
Aditya Rahi (2020MT10784)

F = ∑(0,1,2,5,6,8,9,11,13,14,15)
The **truth table** is:

A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Implementation methodology and explanation:
Considering B, C and D as select lines for the multiplexer, we can observe the variation of F with A to obtain the desired realisation using minimum additional gates.

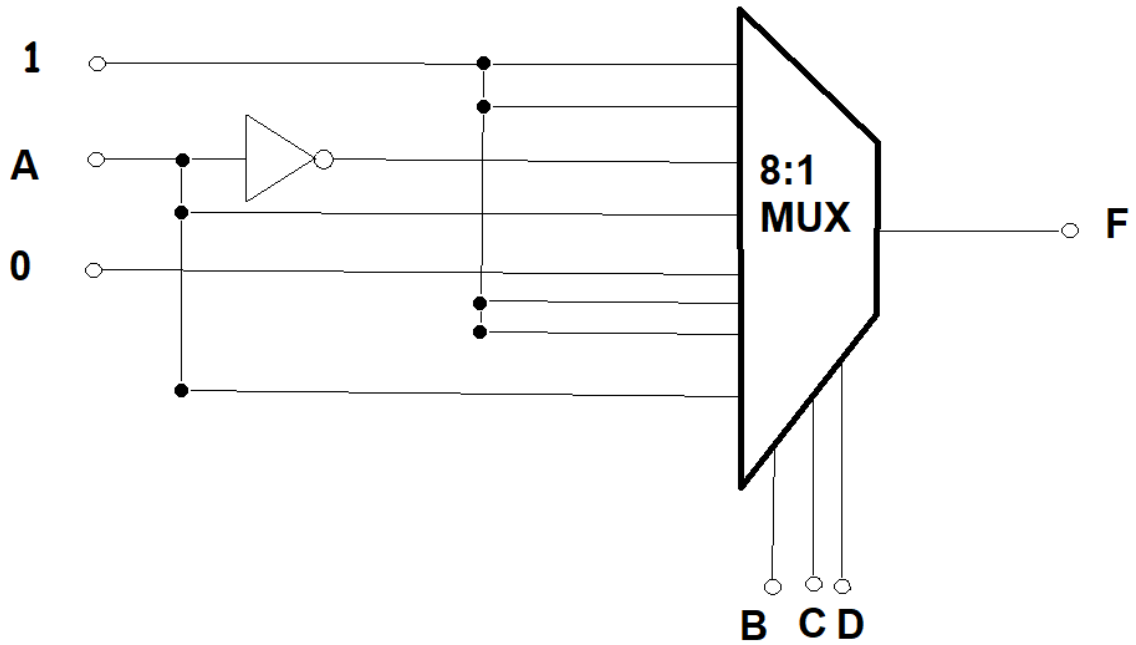
Consider the following table:

	D0	D1	D2	D3	D4	D5	D6	D7
A’	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	1	1	A’	A	0	1	1	A

Here, D0, D1, D2, …, D7 are input signals for the multiplexer. The main inputs will be 0, 1, A and A’. The highlighted elements in the above table shows the inputs where F is 1.

This implementation can be done by:
i) a single 8 X 1 Multiplexer
ii) two 4 X 1 Multiplexers

Circuit Diagram:



Verilog Code:

(8 X 1 Multiplexer)

```
// Module named "MUX8to1" with 8 input lines, 3 select lines and 1 output port
```

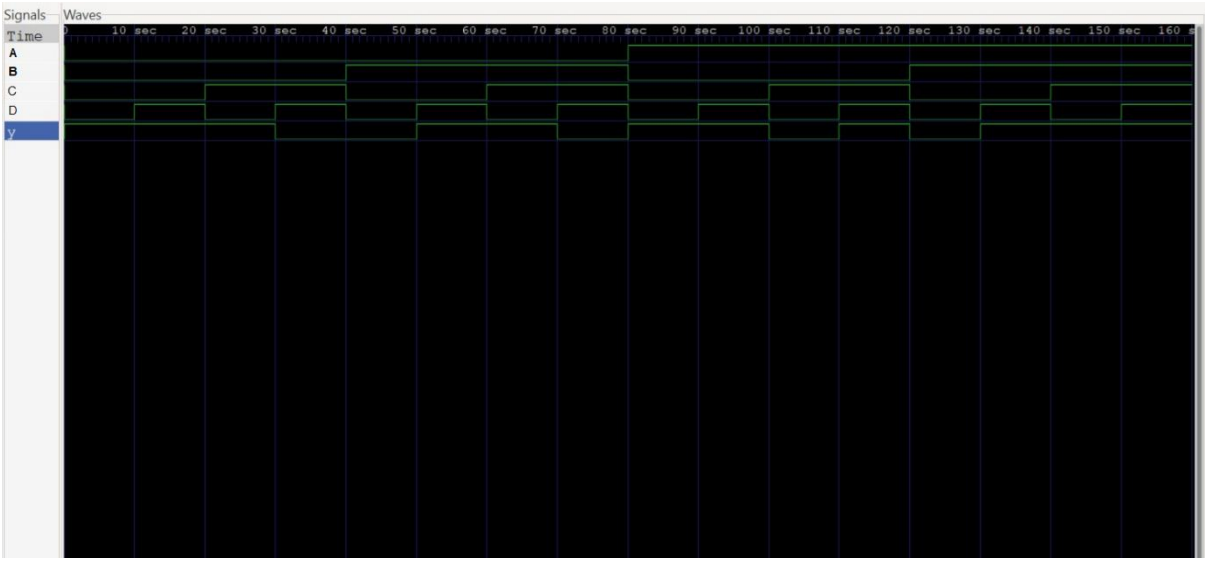
```
module MUX8to1 (input d0, d1, d2, d3, d4, d5, d6, d7, s0, s1, s2, // input
signals and select lines
                output reg F // one output
        );
// Contents of the module
always @ (d0, d1, d2, d3, d4, d5, d6, d7, s0, s1, s2, F)  begin
        if(s2 == 0 && s1 == 0 && s0 == 0)   out = d0;
    else if(s2 == 0 && s1 == 0 && s0 == 1)  out = d1;
    else if(s2 == 0 && s1 == 1 && s0 == 0)  out = d2;
    else if(s2 == 0 && s1 == 1 && s0 == 1)  out = d3;
    else if(s2 == 1 && s1 == 0 && s0 == 0)  out = d4;
    else if(s2 == 1 && s1 == 0 && s0 == 1)  out = d5;
    else if(s2 == 1 && s1 == 1 && s0 == 0)  out = d6;
    else if(s2 == 1 && s1 == 1 && s0 == 1)  out = d7; // Select lines inputs
linked with the corresponding output
    end
endmodule
// Module named "solution" for our required inputs A,B,C and D
module solution(input A, B, C, D, output Y);
MUX8to1 sol(1'b1,1'b1,~A,A,1'b0,1'b1,1'b1,A,D,C,B,Y);
endmodule;
```

Verilog Testbench Code:

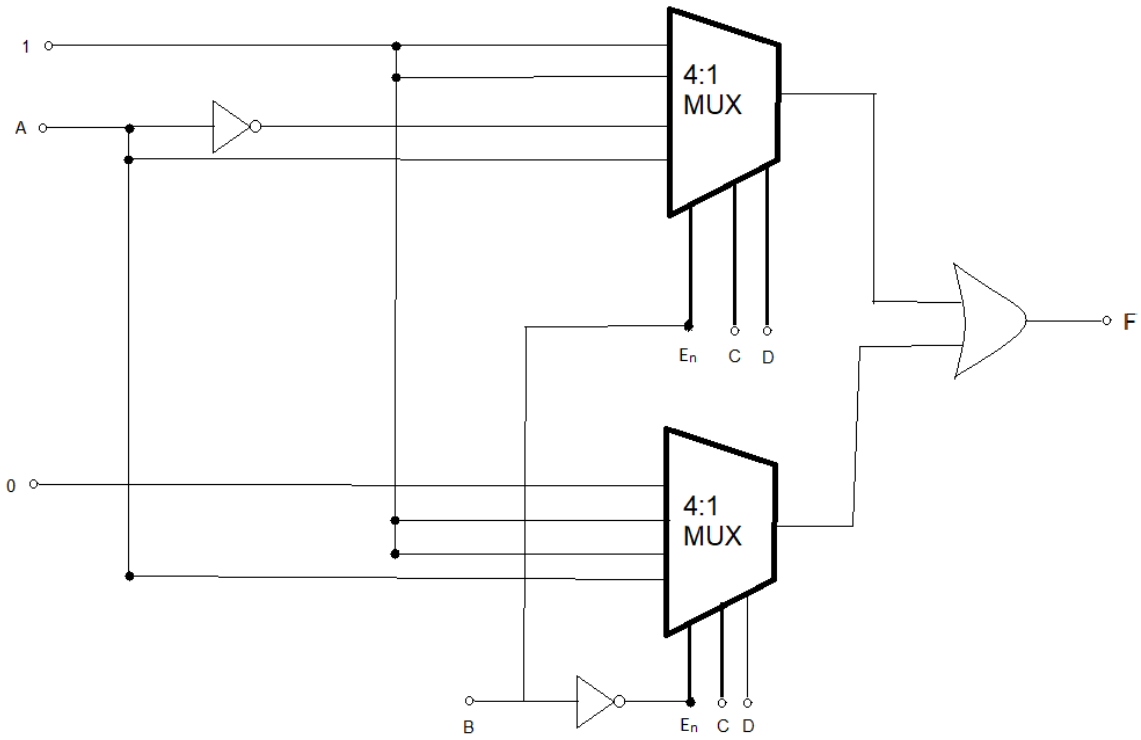
```
`timescale 1s/100ms
`include "mux8to1.v"
module mux8to1_tb; //Test module
    // Inputs
    reg A;
    reg B;
    reg C;
    reg D;
    // Output
    wire out;
    solution check(A, B, C, D, y); // Instantiating the solution under test
    initial begin
        //initialise inputs
        A = 0; B = 0; C = 0; D = 0;
        $monitor("A=%b, B=%b, C=%b, D=%b, out=%b",A,B,C,D,y);
        $dumpfile("mux8to1.vcd");
        $dumpvars(0,MUX8to1_tb);
        A = 0; B = 0; C = 0; D = 0; #10;
        A = 0; B = 0; C = 0; D = 1; #10;
        A = 0; B = 0; C = 1; D = 0; #10;
        A = 0; B = 0; C = 1; D = 1; #10;
        A = 0; B = 1; C = 0; D = 0; #10;
        A = 0; B = 1; C = 0; D = 1; #10;
```

```
        A = 0; B = 1; C = 1; D = 0; #10;
        A = 0; B = 1; C = 1; D = 1; #10;
        A = 1; B = 0; C = 0; D = 0; #10;
        A = 1; B = 0; C = 0; D = 1; #10;
        A = 1; B = 0; C = 1; D = 0; #10;
        A = 1; B = 0; C = 1; D = 1; #10;
        A = 1; B = 1; C = 0; D = 0; #10;
        A = 1; B = 1; C = 0; D = 1; #10;
        A = 1; B = 1; C = 1; D = 0; #10;
        A = 1; B = 1; C = 1; D = 1; #10;
        $finish; // finish stimulation
    end
endmodule
```

Testbench Waveform:



Circuit Diagram (for 4 X 1 Multiplexer):

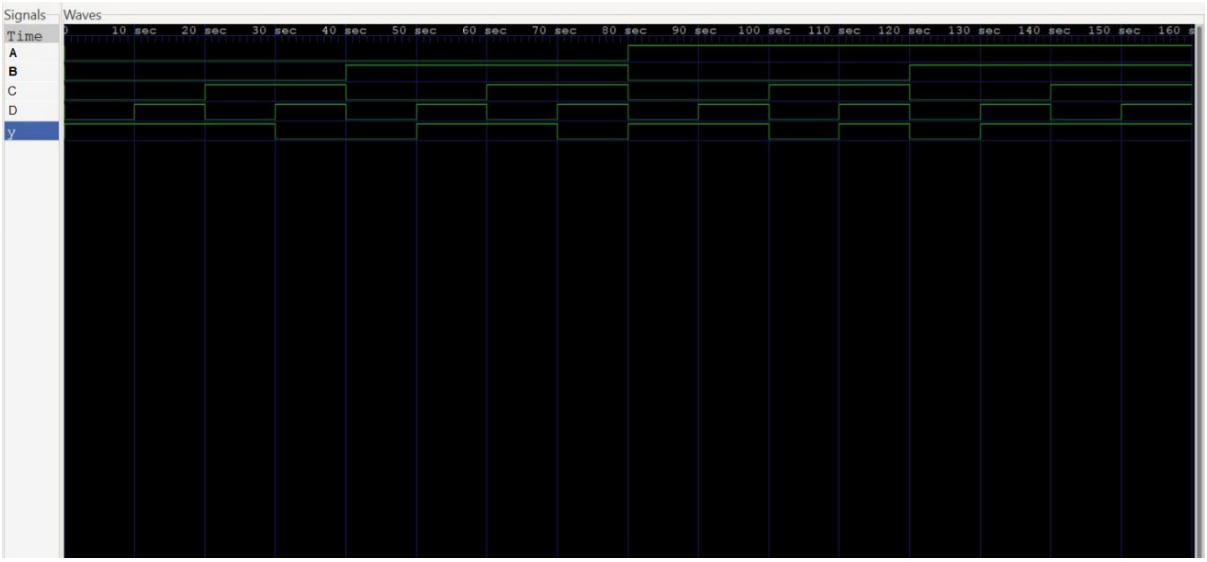


Verilog Code:

(4 X 1 Multiplexer)

```
module MUX4to1 (
    input a0, a1, a2, a3, s0, s1,
    input en,
    output reg out
);
    always @ (a0, a1, a2, a3, s0, s1)
    if(en == 1) out = 0;
    else
    out = (a0 & ~s1 & ~s0) | (a1 & ~s1 & s0) | (a2 & s1 & ~s0) | (a3 & s1 &
s0);
endmodule
module solution( input A, B, C, D
        output y );
    wire w1, w2;
    MUX4to1 upper(1'b1, 1'b1, ~A, A, D, C, B, w1);
    MUX4to1 lower(1'b0, 1'b1, 1'b1, A, D, C, ~B, w2);
    assign y = upper | lower;
endmodule
```

Testbench Waveform:



Implementation methodology:

Here, the basic idea behind implementation is usage of two 4 X 1 multiplexers as a single 8 X 1 multiplexer. This is done by using B as an enable signal, i.e., if B = 0, the upper 4 X 1 multiplexer gives the desired output, since the **enable pin is Active LOW.** Similarly, if B = 1, the lower multiplexer works.