ELL201: DIGITAL ELECTRONICS
ASSIGNMENT 1
VERILOG

Ritika Soni

2020MT10838

24th Feb'22

. The function, F(A,B,C,D) = ∑(0,1,2,5,6,8,9,11,13,14,15) is realized using 8 to 1 multiplexers (74LS151), and minimum additional gates.

Truth Table:

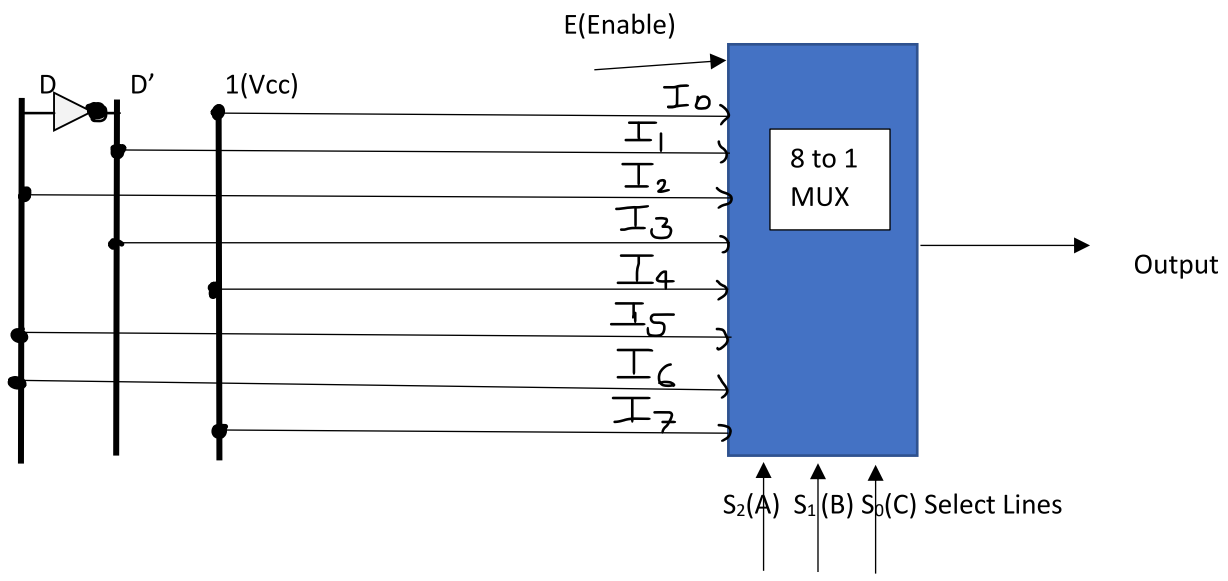
A(S2)	B(S1)	C(S0)	D	F	
0	0	0	0	1	} I_0
0	0	0	1	1	
0	0	1	0	1	} I_1
0	0	1	1	0	
0	1	0	0	0	} I_2
0	1	0	1	1	
0	1	1	0	1	} I_3
0	1	1	1	0	
1	0	0	0	1	} I_4
1	0	0	1	1	
1	0	1	0	0	} I_5
1	0	1	1	1	
1	1	0	0	0	} I_6
1	1	0	1	1	
1	1	1	0	1	} I_7
1	1	1	1	1	

Here S0, S1 , S2 represent select lines.

Observing the above table, we concluded that:

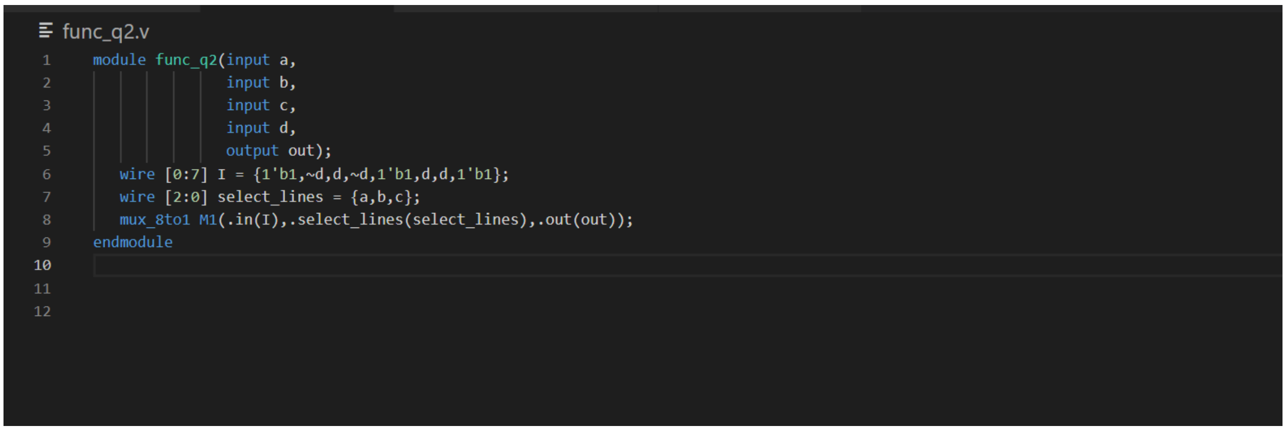
I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
1	~D	D	~D	1	D	D	1

Circuit diagram of the given function is :



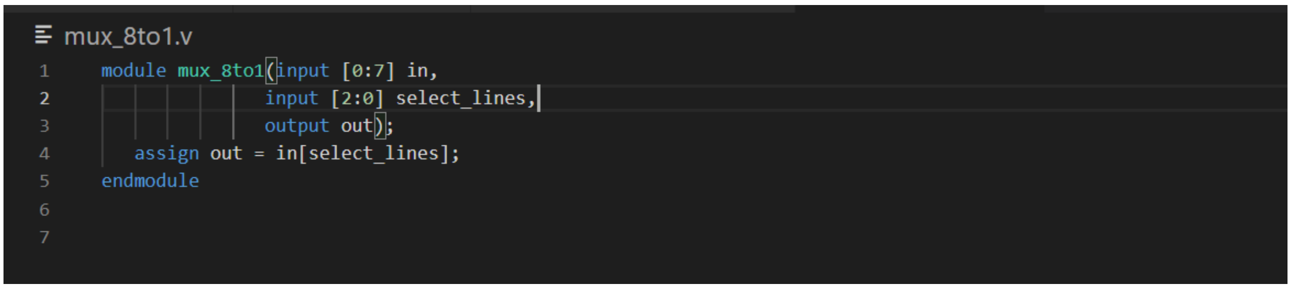
<u>Procedure:</u> Here in the given function we are provided with input variables (A,B,C,D).So in order to implement the 8to1 mux we need 8 inputs, hence we have merged inputs(shown in yellow color in the table) having the same values of A,B, and C namely I_0, I_1 etc. The value of D will finally determine our output of each same input provided by I_0, I_1 etc. We have then found the output of the function in terms of D as shown in the table . Using the table above we have created our circuit diagram having 8 inputs and one output . In our implementation of 8to1 mux , we have taken A , B, C as our S2,S1,S0 select lines to select our input .Inputs get their values depending on values of D using our table.

Code for Function F and 8to1mux :



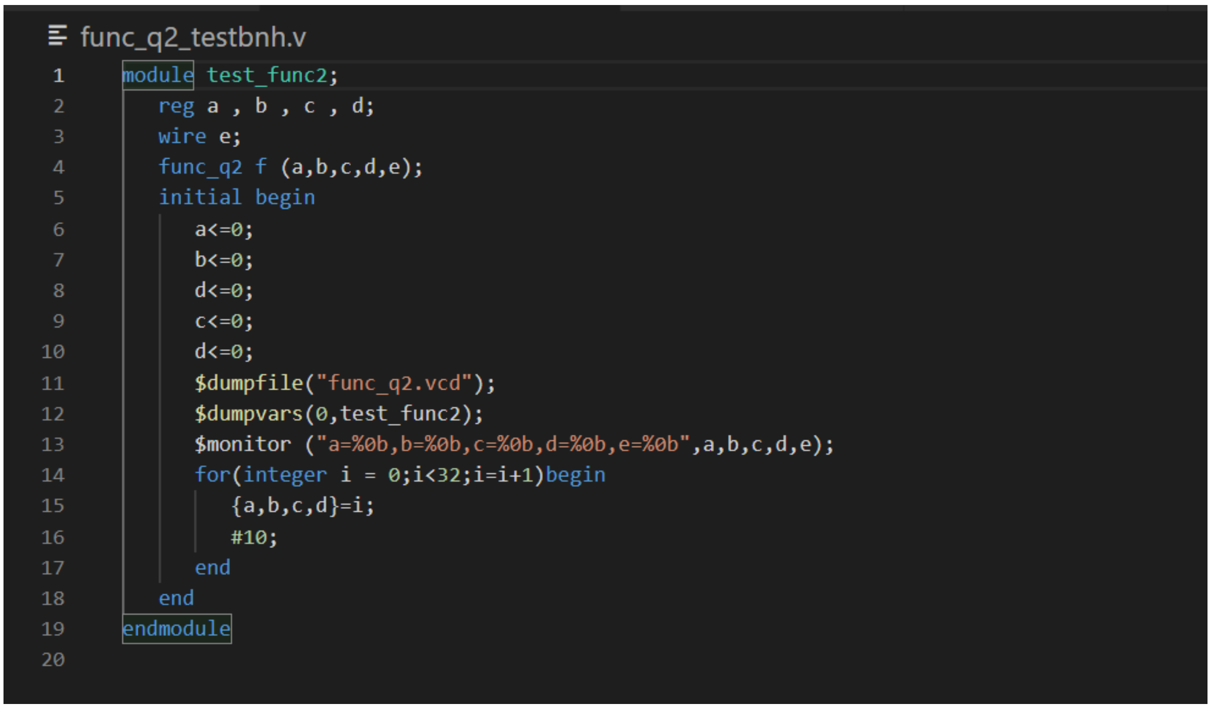
func_q2.v

```
module func_q2(input a,
               input b,
               input c,
               input d,
               output out);
    wire [0:7] I = {1'b1,~d,d,~d,1'b1,d,d,1'b1};
    wire [2:0] select_lines = {a,b,c};
    mux_8to1 M1(.in(I),.select_lines(select_lines),.out(out));
endmodule
```



mux_8to1.v

```
module mux_8to1(input [0:7] in,
                input [2:0] select_lines,
                output out);
    assign out = in[select_lines];
endmodule
```

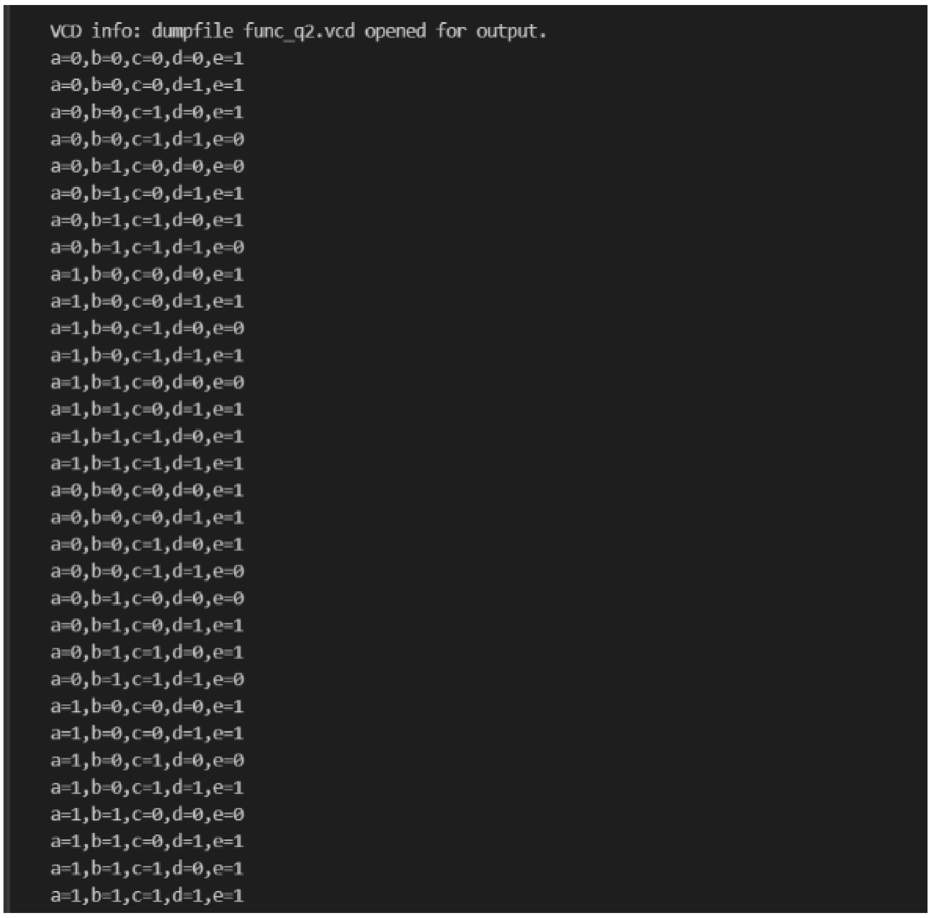


func_q2_testbnh.v

```
module test_func2;
    reg a , b , c , d;
    wire e;
    func_q2 f (a,b,c,d,e);
    initial begin
        a<=0;
        b<=0;
        d<=0;
        c<=0;
        d<=0;
        $dumpfile("func_q2.vcd");
        $dumpvars(0,test_func2);
        $monitor ("a=%0b,b=%0b,c=%0b,d=%0b,e=%0b",a,b,c,d,e);
        for(integer i = 0;i<32;i=i+1)begin
            {a,b,c,d}=i;
            #10;
        end
    end
endmodule
```

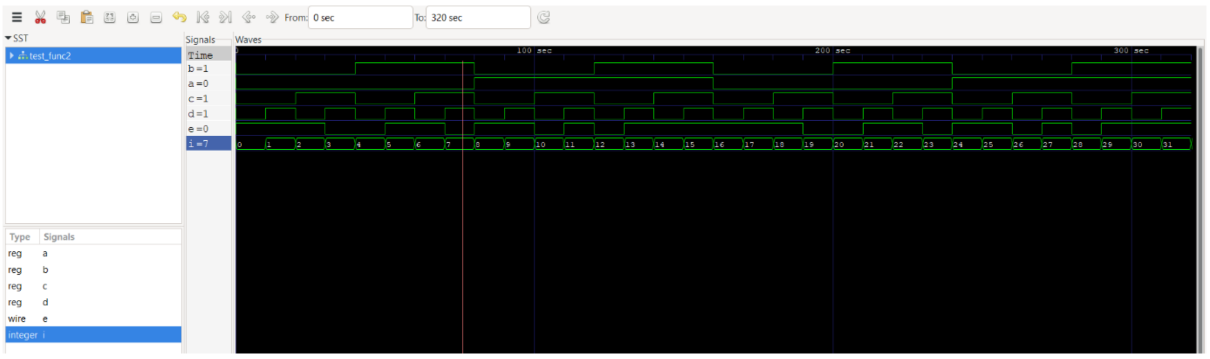
<u>Explaination of Code:</u> I have first created a function that takes a, b , c ,d as inputs and returns the output e(our function value). This function directly uses our function of 8to1 mux which takes 8 inputs and returns one output. The 8to1 mux module takes 8 input lines and select lines as input and outputs the function value. The test bench provided here takes all possible values of a , b , c , d and provides output,i.e., e. 16 values for abcd are sufficient to provide the required output in the test bench (loop given in the end from 0 to 32) but I have taken according to my will for more test cases.

Output of testbench: e is the output of function



```
VCD info: dumpfile func_q2.vcd opened for output.
a=0,b=0,c=0,d=0,e=1
a=0,b=0,c=0,d=1,e=1
a=0,b=0,c=1,d=0,e=1
a=0,b=0,c=1,d=1,e=0
a=0,b=1,c=0,d=0,e=0
a=0,b=1,c=0,d=1,e=1
a=0,b=1,c=1,d=0,e=1
a=0,b=1,c=1,d=1,e=0
a=1,b=0,c=0,d=0,e=1
a=1,b=0,c=0,d=1,e=1
a=1,b=0,c=1,d=0,e=0
a=1,b=0,c=1,d=1,e=1
a=1,b=1,c=0,d=0,e=0
a=1,b=1,c=0,d=1,e=1
a=1,b=1,c=1,d=0,e=1
a=1,b=1,c=1,d=1,e=1
a=0,b=0,c=0,d=0,e=1
a=0,b=0,c=0,d=1,e=1
a=0,b=0,c=1,d=0,e=1
a=0,b=0,c=1,d=1,e=0
a=0,b=1,c=0,d=0,e=0
a=0,b=1,c=0,d=1,e=1
a=0,b=1,c=1,d=0,e=1
a=0,b=1,c=1,d=1,e=0
a=1,b=0,c=0,d=0,e=1
a=1,b=0,c=0,d=1,e=1
a=1,b=0,c=1,d=0,e=0
a=1,b=0,c=1,d=1,e=1
a=1,b=1,c=0,d=0,e=0
a=1,b=1,c=0,d=1,e=1
a=1,b=1,c=1,d=0,e=1
a=1,b=1,c=1,d=1,e=1
```

Output of Gtkwave:



Q2. The function, F(A,B,C,D) = ∑(0,1,2,5,6,8,9,11,13,14,15) is realized using two 4 to 1 multiplexers (74LS151), and minimum additional gates.

Truth Table :

A	B(S2)	C(S1)	D(S0)	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

Here S0 , S1 , S2 represents select lines.

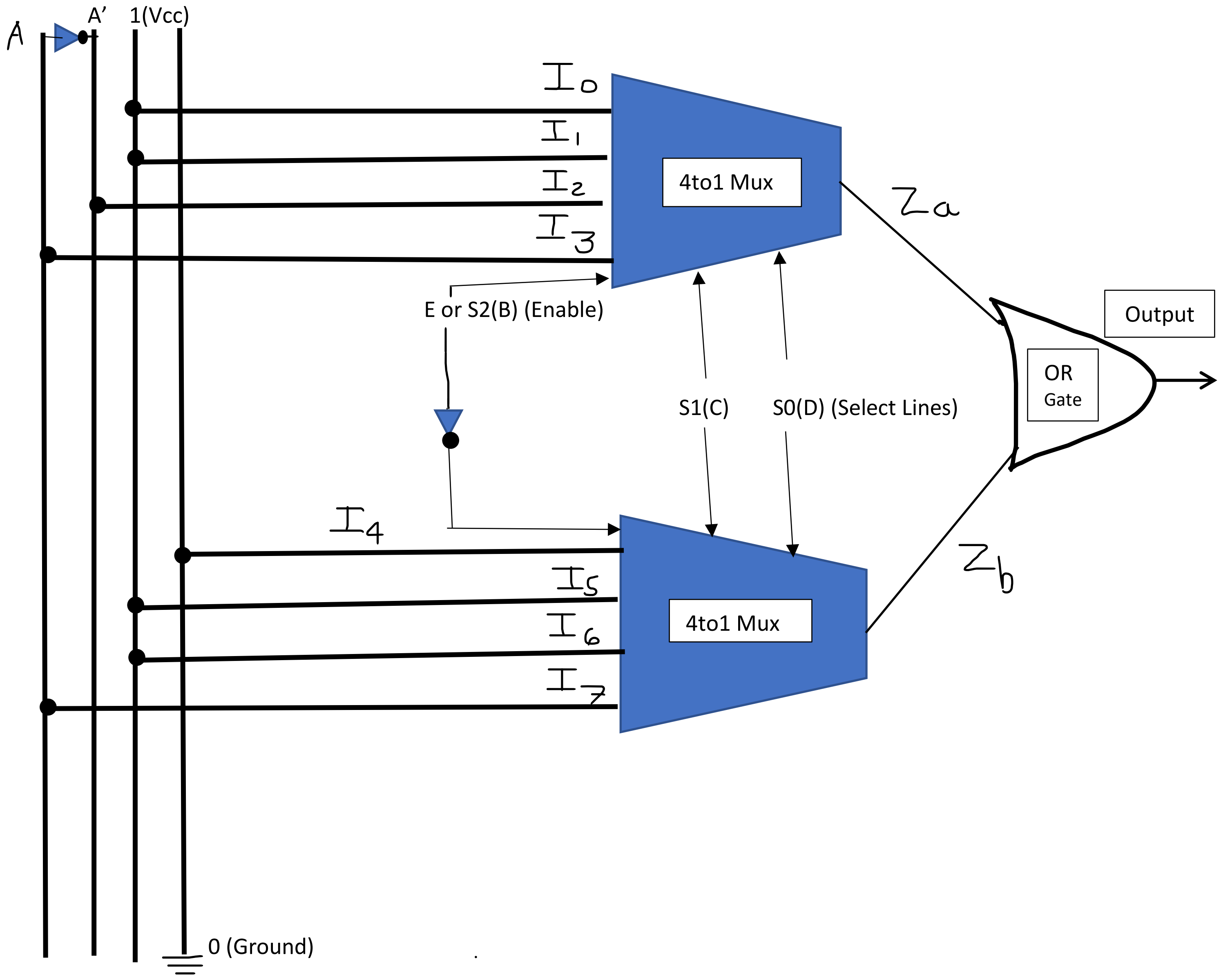
	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{A}$	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	1	1	$\bar{A}$	A	0	1	1	A

Procedure:

Here we are given a function that takes 4 input variables namely A , B , C and D. We have taken all possible values of the combination of these variables taking an as MSB and D as LSB. Then we constructed our table having a value of function 1 wherever the minterms are 1 as given in the question. Now we have constructed another table where A is equal to 1 and A' is 0 . As we have 16 input values but just 2 4 to 1 mux so we will use this table to reduce the number of inputs to 8. We will list the number of all possible values of inputs as shown in table 2. Then we will mark all those minterm values which make our function 1 with red circle. If both the values in the column are marked red then no matter what is the value of I_X , it will always give true(1) output so we have taken it to be Vcc . Similarly, if both values are not marked with red circle it implies that the value of the function will be false(0), therefore we have connected that I_x to the ground(0). And finally, taken the value of A or A' where the function becomes true(1).

Now we have taken the initial 4 values for the first 4to1 mux and the rest of the values for the other 4to 1 mux. The variable B will be working as enable. While C and D will work as select lines S1 and S0 respectively. the upper /first 4 to 1 mux will give Za as output while the lower /second 4 to 1 mux will give Zb as its output. Finally, we will take the or of both the outputs to produce our final output

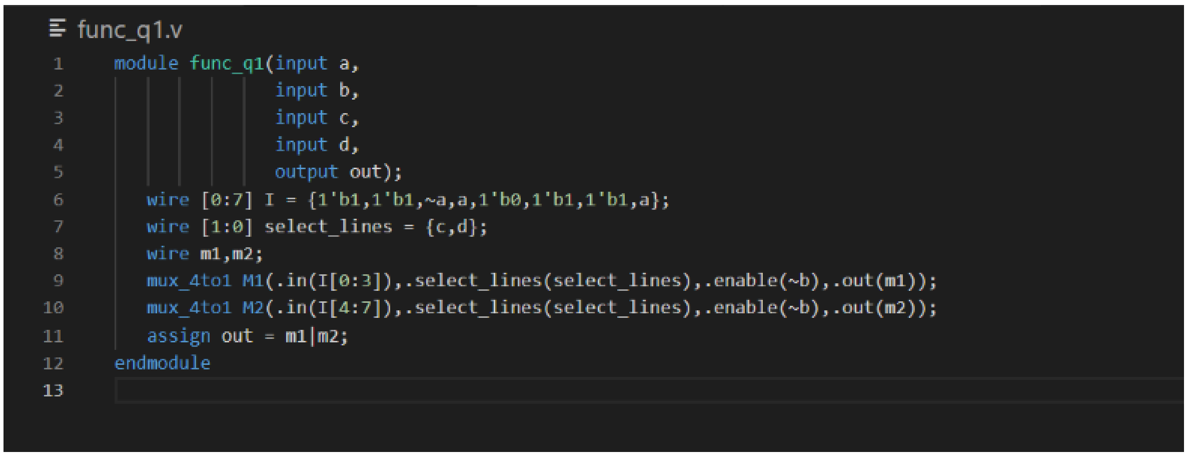
Circuit diagram of 8to1 mux using two 4to1 mux:



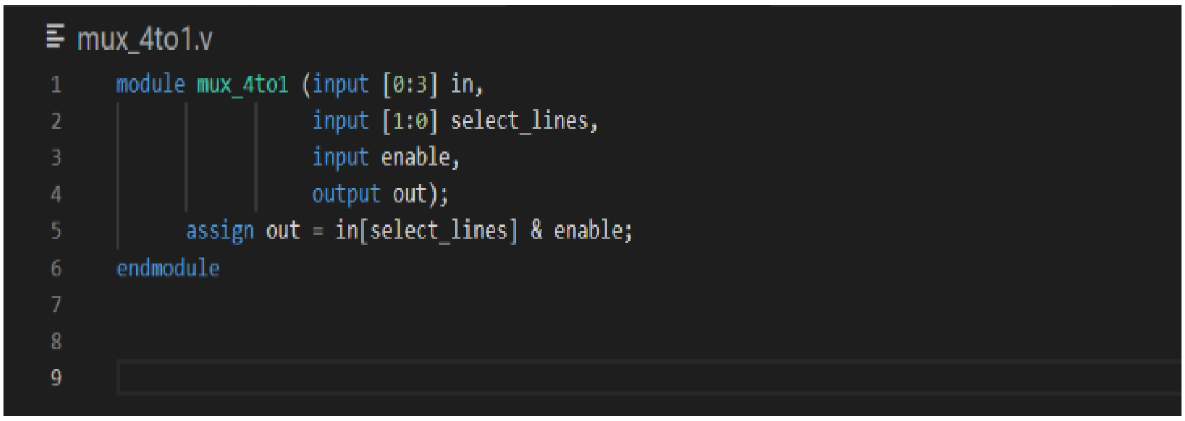
Explanation of Code: Here I have created a function that takes 4 input variables a ,b , c , d , and returns output out. I have used 2 4to1 mux here. First mux takes 4 inputs and returns the output , named m1 in code, of those inputs while the lower 4 to 1 mux takes the rest of the inputs and returns the output of the rest of the 4 inputs, named as m2 in the code below, finally at the end we have combined the inputs by using or operator of both the outputs.

The 4to1 mux takes 4 inputs and returns the value of function according to the value of select lines and enable. . The test bench provided here takes all possible values of a , b , c , d and provides output,i.e., e. 16 values for abcd are sufficient to provide the required output in the test bench (loop given in the end from 0 to 32) but I have taken according to my will for more test cases.

Code for function F and 4to1 mux:

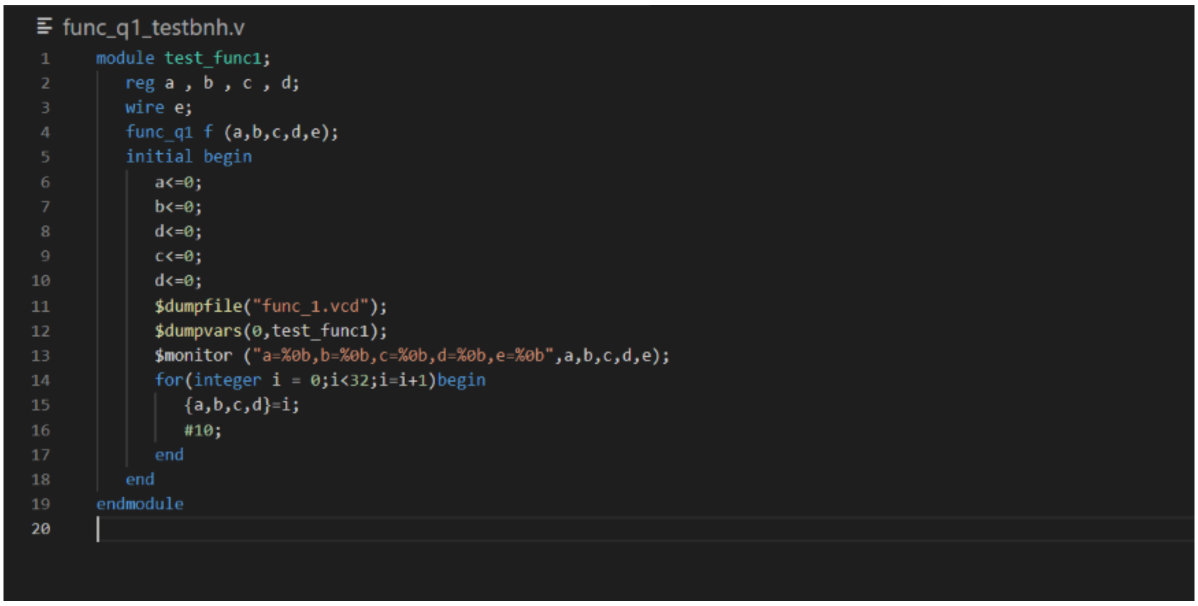


```
module func_q1(input a,
               input b,
               input c,
               input d,
               output out);
   wire [0:7] I = {1'b1,1'b1,~a,a,1'b0,1'b1,1'b1,a};
   wire [1:0] select_lines = {c,d};
   wire m1,m2;
   mux_4to1 M1(.in(I[0:3]),.select_lines(select_lines),.enable(~b),.out(m1));
   mux_4to1 M2(.in(I[4:7]),.select_lines(select_lines),.enable(~b),.out(m2));
   assign out = m1|m2;
endmodule
```



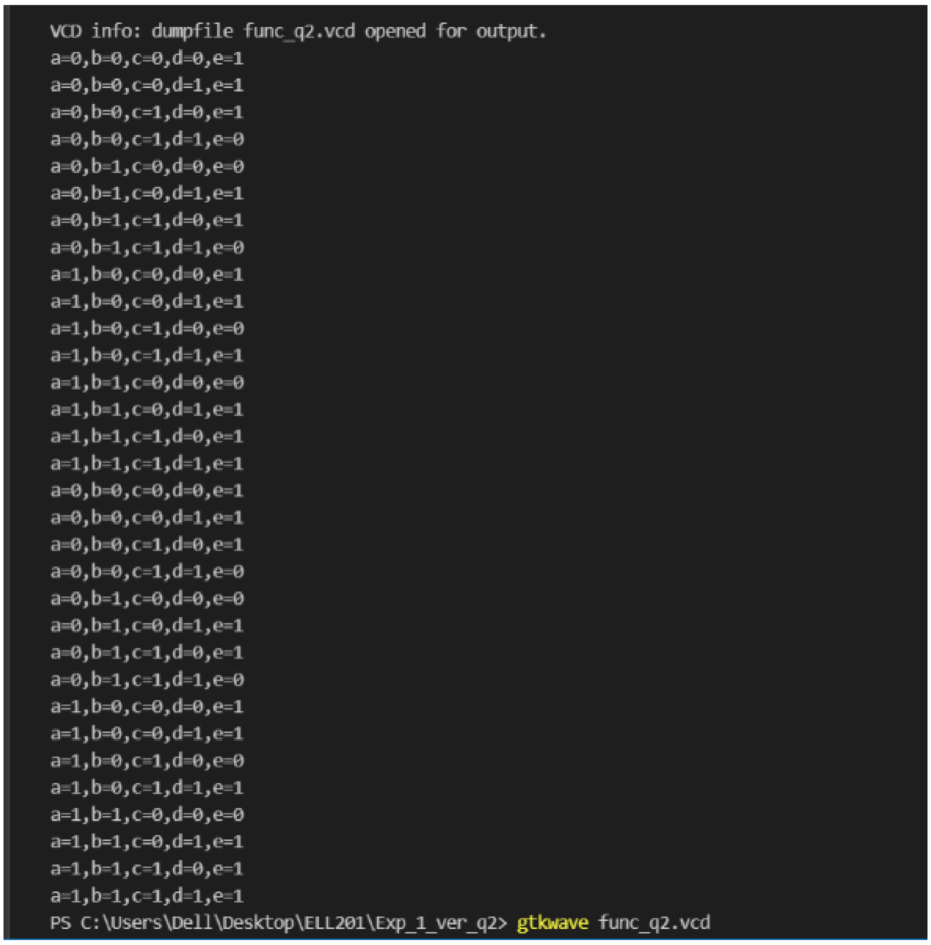
```
module mux_4to1 (input [0:3] in,
                 input [1:0] select_lines,
                 input enable,
                 output out);
      assign out = in[select_lines] & enable;
endmodule
```

Code for Testbench:



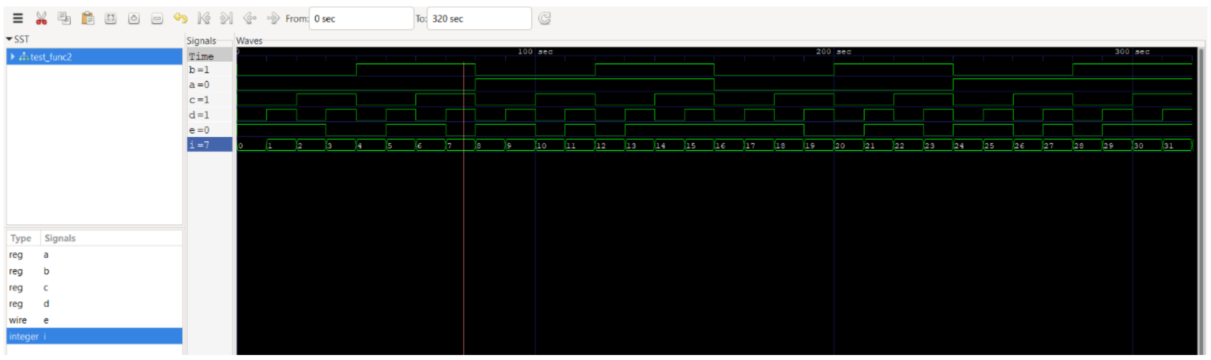
```
module test_func1;
   reg a , b , c , d;
   wire e;
   func_q1 f (a,b,c,d,e);
   initial begin
      a<=0;
      b<=0;
      d<=0;
      c<=0;
      d<=0;
      $dumpfile("func_1.vcd");
      $dumpvars(0,test_func1);
      $monitor ("a=%0b,b=%0b,c=%0b,d=%0b,e=%0b",a,b,c,d,e);
      for(integer i = 0;i<32;i=i+1)begin
         {a,b,c,d}=i;
         #10;
      end
   end
endmodule
```

Output of Testbench: e is the output of function



```
VCD info: dumpfile func_q2.vcd opened for output.
a=0,b=0,c=0,d=0,e=1
a=0,b=0,c=0,d=1,e=1
a=0,b=0,c=1,d=0,e=1
a=0,b=0,c=1,d=1,e=0
a=0,b=1,c=0,d=0,e=0
a=0,b=1,c=0,d=1,e=1
a=0,b=1,c=1,d=0,e=1
a=0,b=1,c=1,d=1,e=0
a=1,b=0,c=0,d=0,e=1
a=1,b=0,c=0,d=1,e=1
a=1,b=0,c=1,d=0,e=0
a=1,b=0,c=1,d=1,e=1
a=1,b=1,c=0,d=0,e=0
a=1,b=1,c=0,d=1,e=1
a=1,b=1,c=1,d=0,e=1
a=1,b=1,c=1,d=1,e=1
a=0,b=0,c=0,d=0,e=1
a=0,b=0,c=0,d=1,e=1
a=0,b=0,c=1,d=0,e=1
a=0,b=0,c=1,d=1,e=0
a=0,b=1,c=0,d=0,e=0
a=0,b=1,c=0,d=1,e=1
a=0,b=1,c=1,d=0,e=1
a=0,b=1,c=1,d=1,e=0
a=1,b=0,c=0,d=0,e=1
a=1,b=0,c=0,d=1,e=1
a=1,b=0,c=1,d=0,e=0
a=1,b=0,c=1,d=1,e=1
a=1,b=1,c=0,d=0,e=0
a=1,b=1,c=0,d=1,e=1
a=1,b=1,c=1,d=0,e=1
a=1,b=1,c=1,d=1,e=1
PS C:\Users\Dell\Desktop\ELL201\Exp_1_ver_q2> gtkwave func_q2.vcd
```

Output of Gtkwave:


From: 0 sec
To: 320 sec
SST
test_func2
Signals
Waves
Time
b =1
a =0
c =1
d =1
e =0
i =7
100 sec
200 sec
300 sec
Type Signals
reg a
reg b
reg c
reg d
wire e
integer i