HACETTEPE UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BBM 233 LOGIC DESIGN LAB – 2021 FALL

Lab Experiment 4 Verilog Combinational

December 5, 2021

Student name: Umut GÜNGÖR

Student Number: b21946198

1 Problem Definition

We are asked to design a circuit which has given function with usign 2x4 decoder and 4x1 multiplexer in Verilog. We write our codes and test benches for testing our code in waveforms. To connect decoder and multiplexer we need some instantiations.

2 Solution Implementation

2.1 2x4 Decoder Implementation

To make 2x4 decoder, we need 2 inputs to obtain 4 outputs. These inputs come from function F(a,b,c,d)'s parameters a and b. For output options for example one of them is ab which is 11 in binary and 3 in decimal.

decoder_2x4.v

```
1   module decoder_2x4
2   (
3       input[1:0] A,       //2 inputs
4       output[3:0] D       //4 outputs as usual in 2x4 decoder
5
6   );
7       //Used dataflow design approach here
8       assign D[0] = !A[1] & !A[0];
9       assign D[1] = !A[1] & A[0];
10      assign D[2] = A[1] & !A[0];
11      assign D[3] = A[1] & A[0];
12
13
14  endmodule
15
```

3.1 2x4 Decoder Testbench Implementation

We are testing for every binary number which are 00 to 11. For all input values the only one output value is high.

decoder_2x4_tb.v

```
1   `timescale 1ns / 1ps
2   module decoder_2x4_tb;
3       reg [1:0] A;
4       wire [3:0] D;
5
6
7   decoder_2x4 uut(.A(A), .D(D));          //uut function for decoder
8
9
10  initial begin
11      $dumpfile("simulation.vcd");        //for simulation
12      $dumpvars;                          //for pass the variables
13
14      A[1] = 0; A[0] = 0;
15      #10 A[1] = 0; A[0] = 1;
16      #10 A[1] = 1; A[0] = 0;             //all input possibilities with 10ns delay
17      #10 A[1] = 1; A[0] = 1;
18      #10;
19  end
20
21  endmodule
22
```

This figure shows us to how the 2x4 decoder is implemented visually. For all 4 inputs only the one output is high. For instance when A[1] = 1 and A[0] is zero D[2] is 1 and others are 0.

2.2 4x1 Multiplexer Implementation

To make 4x1 multiplexer we need 4 inputs 2 selectors to obtain 1 special output which is chosen by selectors. These inputs comes from decoder's outputs and selectors comes from function F(a,b,c,d)' parameters c and d.

mux_4x1.v

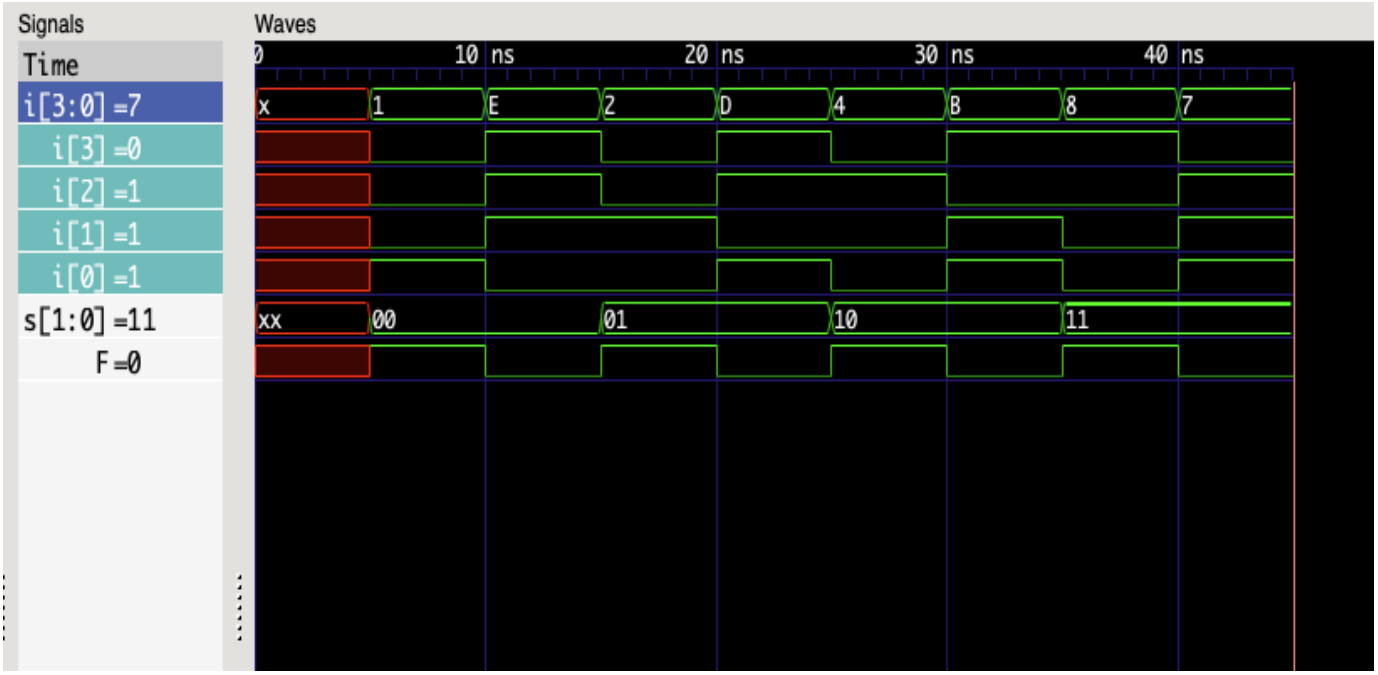
```
1   module mux_4x1
2   (
3       input[3:0] i,           //4 inputs
4       input[1:0] s,           //2 selectors
5       output F                //1 output as usual in 4x1 multiplexer
6
7   );
8   //Used datalow design approach here
9   assign F = (!s[1] & !s[0] & i[0]) | (!s[1] & s[0] & i[1]) | (s[1] & !s[0] & i[2]) | (s[1] & s[0] & i[3]);
10
11  endmodule
12
```

3.2 4X1 Multiplexer Testbench Implementation

We don’t test all 64 cases (64 comes from 4 inputs and 2 selectors 2^6 = 64). We use some cases to show whether multiplexer works correctly or not.

mux_4x1_tb.v

```
1   `timescale 1ns / 1ps
2   module mux_4x1_tb;
3       reg [3:0] i;
4       reg [1:0] s;
5       output F;
6
7
8
9   mux_4x1 uut(.i(i), .s(s), .F(F));          //uut function for mux
10
11  initial begin
12      //I cannot manage for loops here because two distinct for loops needed one for inputs another for selectors
13      //So, I just write some cases
14      //The cases which I don't include are shown as don't cares in simulation
15      #5 i[3] = 0; i[2] = 0; i[1] = 0; i[0] = 1; s[1] = 0; s[0] = 0;
16      #5 i[3] = 1; i[2] = 1; i[1] = 1; i[0] = 0; s[1] = 0; s[0] = 0;
17
18      #5 i[3] = 0; i[2] = 0; i[1] = 1; i[0] = 0; s[1] = 0; s[0] = 1;
19      #5 i[3] = 1; i[2] = 1; i[1] = 0; i[0] = 1; s[1] = 0; s[0] = 1;
20
21      #5 i[3] = 0; i[2] = 1; i[1] = 0; i[0] = 0; s[1] = 1; s[0] = 0;
22      #5 i[3] = 1; i[2] = 0; i[1] = 1; i[0] = 1; s[1] = 1; s[0] = 0;
23
24      #5 i[3] = 1; i[2] = 0; i[1] = 0; i[0] = 0; s[1] = 1; s[0] = 1;
25      #5 i[3] = 0; i[2] = 1; i[1] = 1; i[0] = 1; s[1] = 1; s[0] = 1;
26
27      #5;
28  end
29
30
31  endmodule
32
```



This figure shows that for 8 cases multiplexer works correctly. We don't test 64 cases because it would be too long and complex to analysis.

2.3 Circuit Implementation

In this step we have a given function F(a,b,c,d). We can use its parameters such as a, b, c and d to obtain our final circuit. We use them to connect the decoder and the multiplexer

circuit.v

```
1   module circuit
2   (
3       input a,            //decoder's first input
4       input b,            //decoder's second input
5       input c,            //multiplexer's first selector
6       input d,            //multiplexer's second selctor
7       output F
8
9   );
10
11  wire[3:0] w;            //a vector to connect decoder and multiplexer
12  mux_4x1 m1(.i(w), .s({c,d}), .F(F));        //mux initiation
13  decoder_2x4 d1(.A({a,b}), .D(w));           //decoder initiation
14
15
16
17  wire t1, t2, t3, t4;    //four wires to structural design approach
18  //used structural design approach here
19  and(t1, !a, !b, !c, !d);        //m0
20  and(t2, !a, b, !c, d);          //m5
21  and(t3, a, b, c, d);            //m15
22  and(t4, a, !b, c, !d);          //m10
23  or(F3, t1, t2, t3, t4);         //function F
24
25
26  endmodule
27
```

3.3 Circuit Testbench Implementation

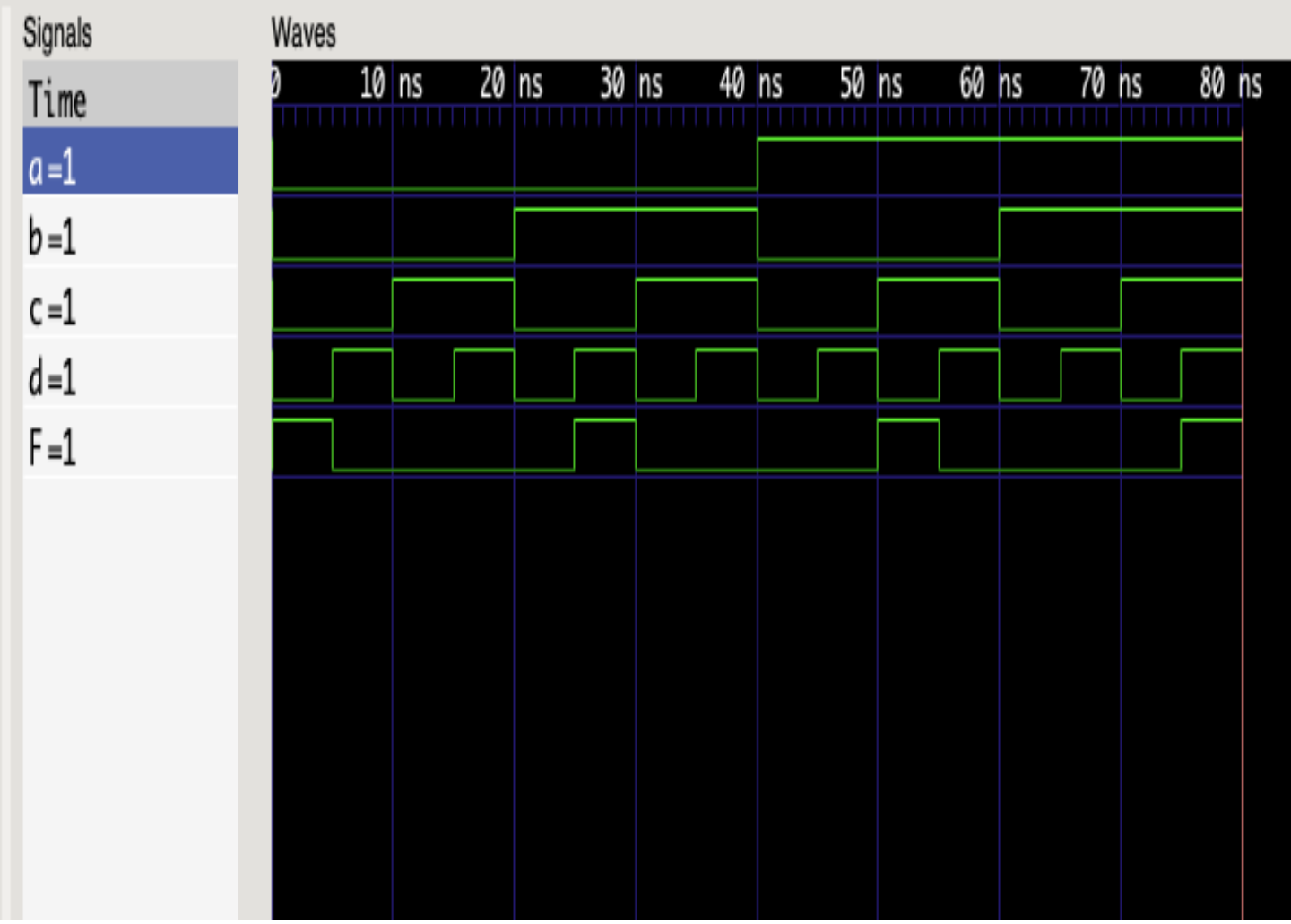
We tests 16 possibilities from 0000 to 1111.

circuit_tb.v

```
1   `timescale 1ns / 1ps
2   module circuit_tb;
3       reg a;
4       reg b;
5       reg c;
6       reg d;
7       output F;
8
9
10  circuit uut(.a(a), .b(b), .c(c), .d(d), .F(F));     //uut function for circuit with explicit association
11
12  initial begin
13      //so here again I cannot manage for loop beacuse there are 4 different inputs
14      a = 0; b = 0; c = 0; d = 0;
15      #5 a = 0; b = 0; c = 0; d = 1;
16      #5 a = 0; b = 0; c = 1; d = 0;
17      #5 a = 0; b = 0; c = 1; d = 1;
18      #5 a = 0; b = 1; c = 0; d = 0;
19      #5 a = 0; b = 1; c = 0; d = 1;
20      #5 a = 0; b = 1; c = 1; d = 0;
21      #5 a = 0; b = 1; c = 1; d = 1;          //all 16 possibilities with 4 inputs
22      #5 a = 1; b = 0; c = 0; d = 0;
23      #5 a = 1; b = 0; c = 0; d = 1;
24      #5 a = 1; b = 0; c = 1; d = 0;
25      #5 a = 1; b = 0; c = 1; d = 1;
26      #5 a = 1; b = 1; c = 0; d = 0;
27      #5 a = 1; b = 1; c = 0; d = 1;
28      #5 a = 1; b = 1; c = 1; d = 0;
29      #5 a = 1; b = 1; c = 1; d = 1;
30      #5;
31  end
32
33
34  endmodule
35
```

4 Results

In conclusion, we understand this combinational circuits are connected to each other. Our final circuit works properly because its formula is F(a,b,c,d) = m0 + m5 + m10 + m15 in minterms form so, it just is only high in these values. It is clearly visible in waveform simulation given below.



6 References

-Some youtube videos
- BBM 231 lecture notes
- BBM 233 Verilog pdf
- www.asic-world.com