Hacettepe University

Computer Engineering Department

BM233 Logic Design Lab - 2021 Fall

Experiment 4 - Combinational Circuits in Verilog

December 4, 2021

Student name:
Mert Ali Yalçin

Student Number:
21946682

1 Problem Definition

Combinational circuits are circuits whose outputs, at any instant of time, depend only on the present inputs (the combinational circuits do not use any memory elements). That is, the previous inputs or state of the circuit do not have any effect on its present state. A combinational circuit performs a specific information-processing operation fully specified logically by a set of Boolean functions. The 'n' input variables come from an external source whereas the 'm' output variables go to an external destination. In many applications, the source or destination are storage registers.

- Decoders
  A decoder is a combinational circuit that converts binary information from n binary inputs to a maximum of 2 n unique output lines. One of these outputs will be active HIGH based on the combination of inputs present, when the decoder is enabled. That means decoder detects a particular code. A decoder provides the 2 n minterms of n input variables
- Multiplexers
  A multiplexer (MUX) is a combinational logic circuit designed to switch one of several input lines through to a single common output line by the application of a control signal. A MUX has a maximum of 2 n data inputs. One of the inputs is connected to the output based on the value of the selection lines. There will be 2 n possible combinations of 1s and 0s since there are 'n' selection lines.
- Combining Components
  Combinational logic circuits can be connected together to form larger circuits. This is done by taking outputs from one circuit and using them as inputs to another. Figure below shows a 2x4 decoder and a 4x1 MUX connected to implement a Boolean function F(a, b, c, d) = $\sum(0.5.10.15)$. 1. In this experiment, we are expected to desing a 2-to-4 Decoder, a 4-to-1 Multiplexer and combine them properly to get the final circuit we want.

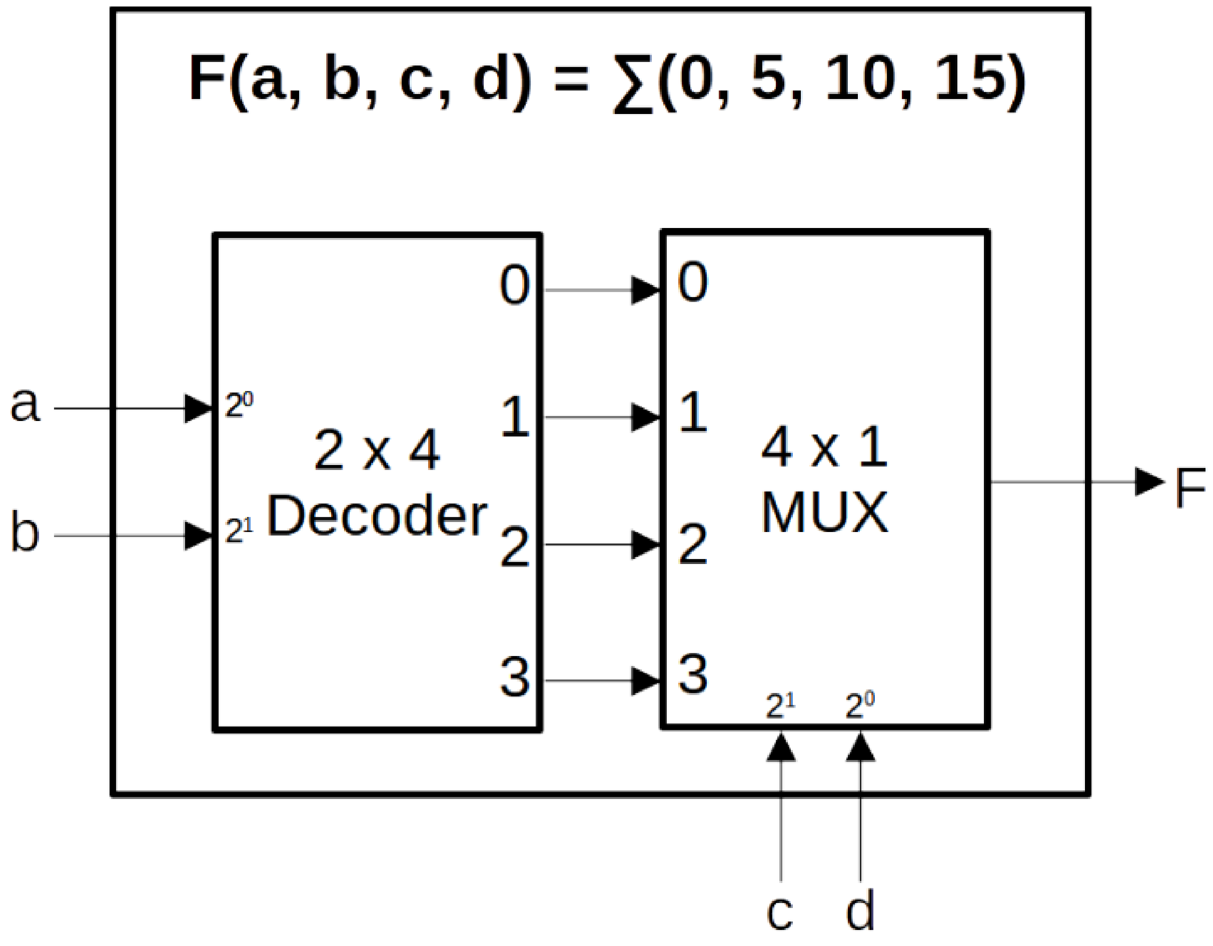

Figure 1: Final Circuit

2 Solution Implementation

decoder_2x4.v

```
module decoder_2x4(
    input[1:0] A,
    output[3:0] D
    );
    assign D[0] = !A[1] && !A[0];
    assign D[1] = !A[1] && A[0];
    assign D[2] = A[1] && !A[0];
    assign D[3] = A[1] && A[0];
endmodule
```

decoder_2x4_tb.v

```
module decoder_2x4_tb;
reg[1:0] A;
wire[3:0] D;
decoder_2x4 UUT(.A(A), .D(D));

initial begin
 A[1] = 0; A[0] = 0;
#5 A[1] = 0; A[0] = 1;
#5 A[1] = 1; A[0] = 0;
#5 A[1] = 1; A[0] = 1;
#5 $finish;
end
endmodule
```

mux_4x1.v

```
module mux_4x1(
    input[3:0] i,
    input[1:0] s,
    output F
    );
    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]);
endmodule
```

mux_4x1_tb.v

```
module mux_4x1_tb;
reg[3:0] i;
reg[1:0] s;
wire F;
mux_4x1 UUT(.i(i), .s(s), .F(F));

```

```
initial begin
// 0 selected
 s[1]=0; s[0]=0; i[3]=0; i[2]=0; i[1]=0; i[0]=1;
#5 s[1]=0; s[0]=0; i[3]=1; i[2]=1; i[1]=1; i[0]=0;
// 1 selected
#5 s[1]=0; s[0]=1; i[3]=0; i[2]=0; i[1]=1; i[0]=0;
#5 s[1]=0; s[0]=1; i[3]=1; i[2]=1; i[1]=0; i[0]=1;
// 2 selected
#5 s[1]=1; s[0]=0; i[3]=0; i[2]=1; i[1]=0; i[0]=0;
#5 s[1]=1; s[0]=0; i[3]=1; i[2]=0; i[1]=1; i[0]=1;
// 3 selected
#5 s[1]=1; s[0]=1; i[3]=1; i[2]=0; i[1]=0; i[0]=0;
#5 s[1]=1; s[0]=1; i[3]=0; i[2]=1; i[1]=1; i[0]=1;
#5 $finish;
end
endmodule
```

circuit.v

```
module circuit(
    input a,
    input b,
    input c,
    input d,
    output F
    );
    wire[3:0] link;
    decoder_2x4 dec(.A({b,a}), .D(link));
    mux_4x1 mux(.i({link[3], link[1], link[2], link[0]}), .s({c,d}), .F(F));
endmodule
```

circuit_tb.v

```
module circuit_tb;
reg a,b,c,d;
wire F;
circuit UUT(.a(a), .b(b), .c(c), .d(d), .F(F));

initial begin
 a=0; b=0; c=0; d=0;
#5 a=0; b=0; c=0; d=1;
#5 a=0; b=0; c=1; d=0;
#5 a=0; b=0; c=1; d=1;
#5 a=0; b=1; c=0; d=0;
#5 a=0; b=1; c=0; d=1;
#5 a=0; b=1; c=1; d=0;
#5 a=0; b=1; c=1; d=1;
#5 a=1; b=0; c=0; d=0;
#5 a=1; b=0; c=0; d=1;
#5 a=1; b=0; c=1; d=0;
```

```
#5 a=1; b=0; c=1; d=1;
#5 a=1; b=1; c=0; d=0;
#5 a=1; b=1; c=0; d=1;
#5 a=1; b=1; c=1; d=0;
#5 a=1; b=1; c=1; d=1;
#5 $finish;
end
endmodule
```

3 Waveforms

decoder 2.

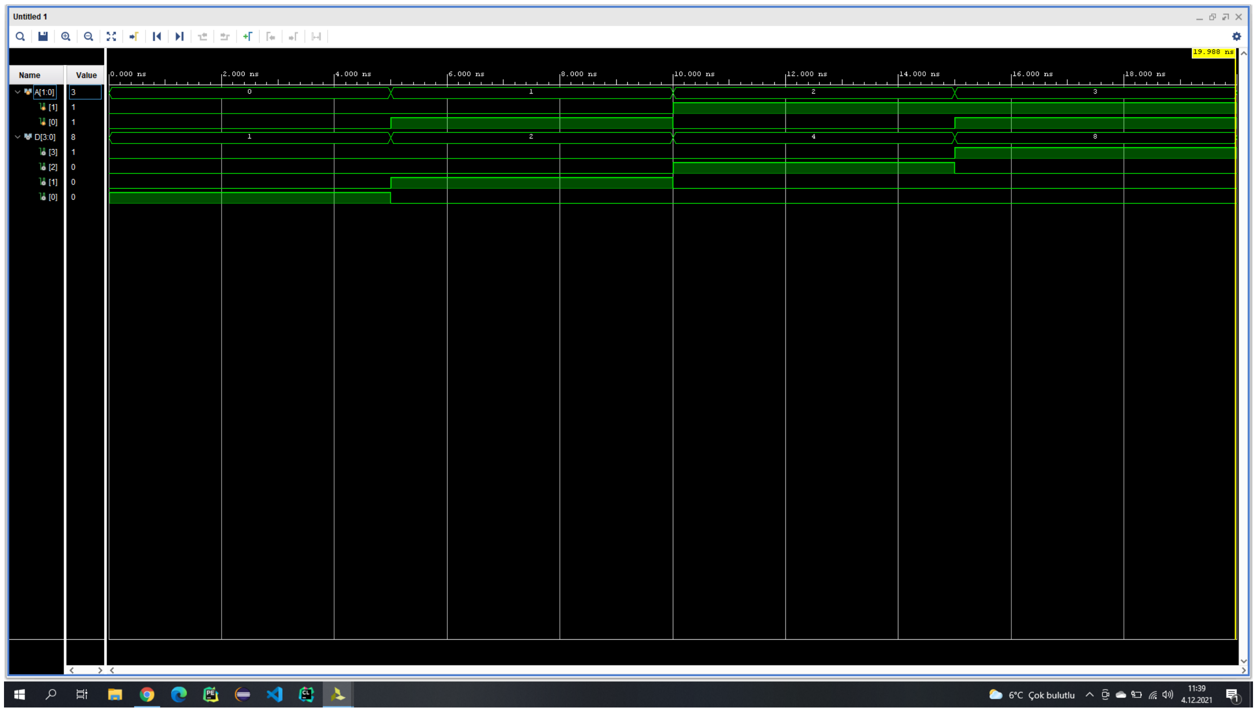

Figure 2: Resulting Waveform of decoder

multiplexer 3.

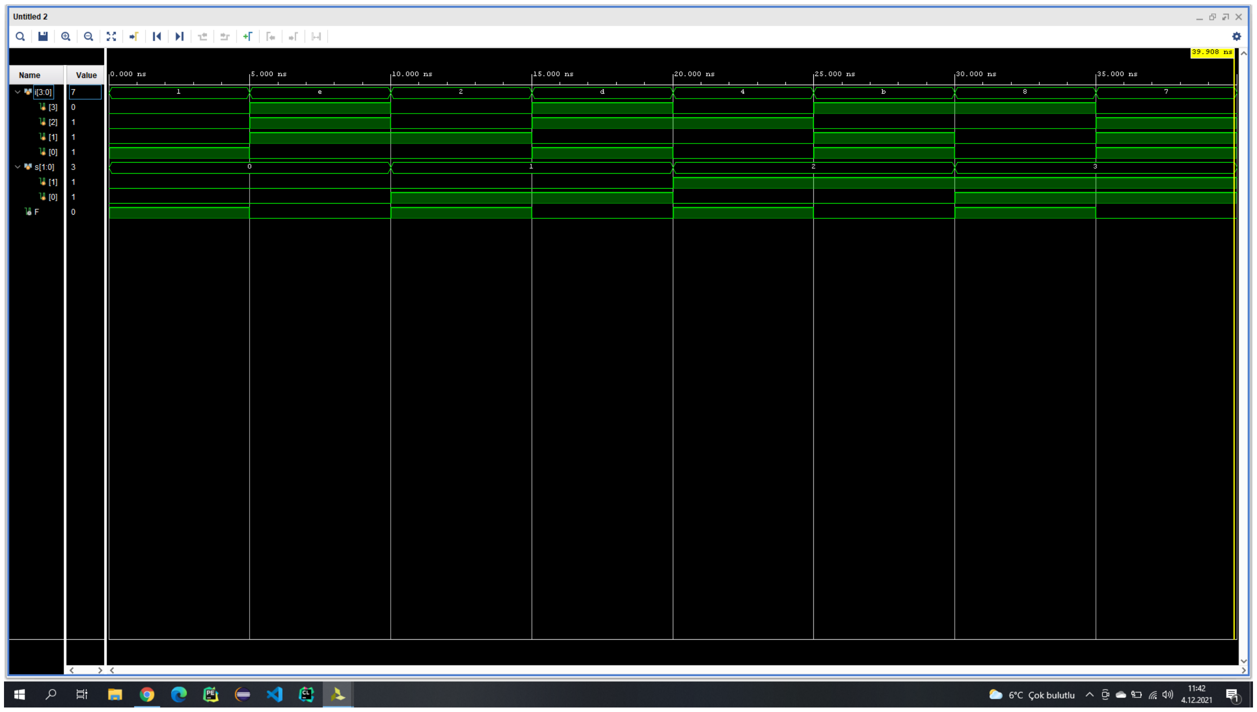

Figure 3: Resulting Waveform of multiplexer

final circuit 4.

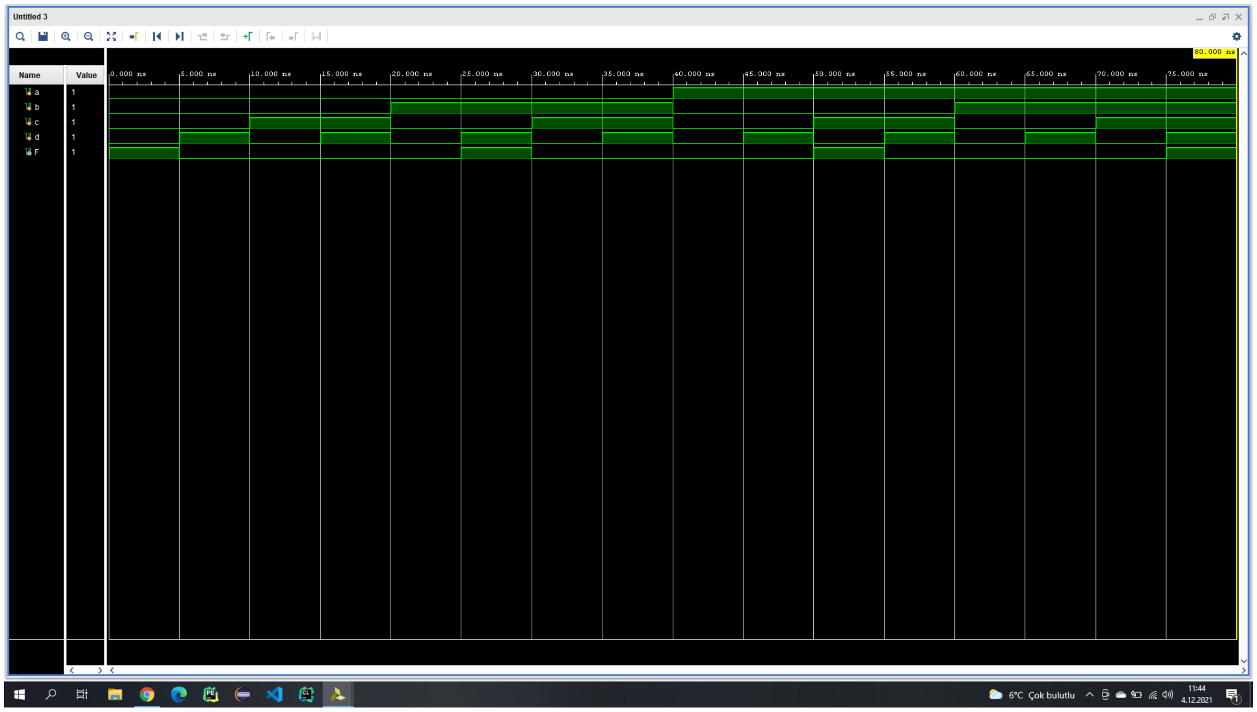

Figure 4: Resulting Waveform of the final circuit

4 Results

decoder 5.
As you can see below, only one output is high in the outputs of the decoder. The high one is selected according to the values of A's.

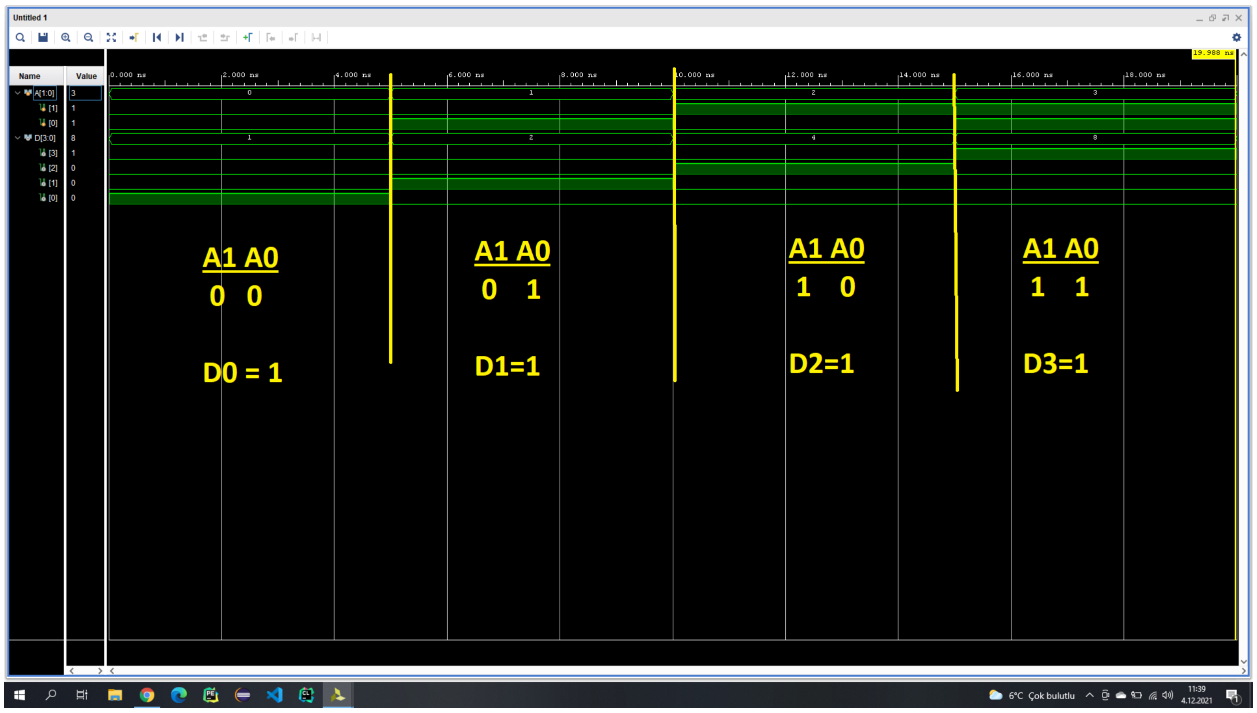

Figure 5: Waveform of decoder explained

multiplexer 6.

In the picture below, the output is equal to the input that I selected via s1 and s2

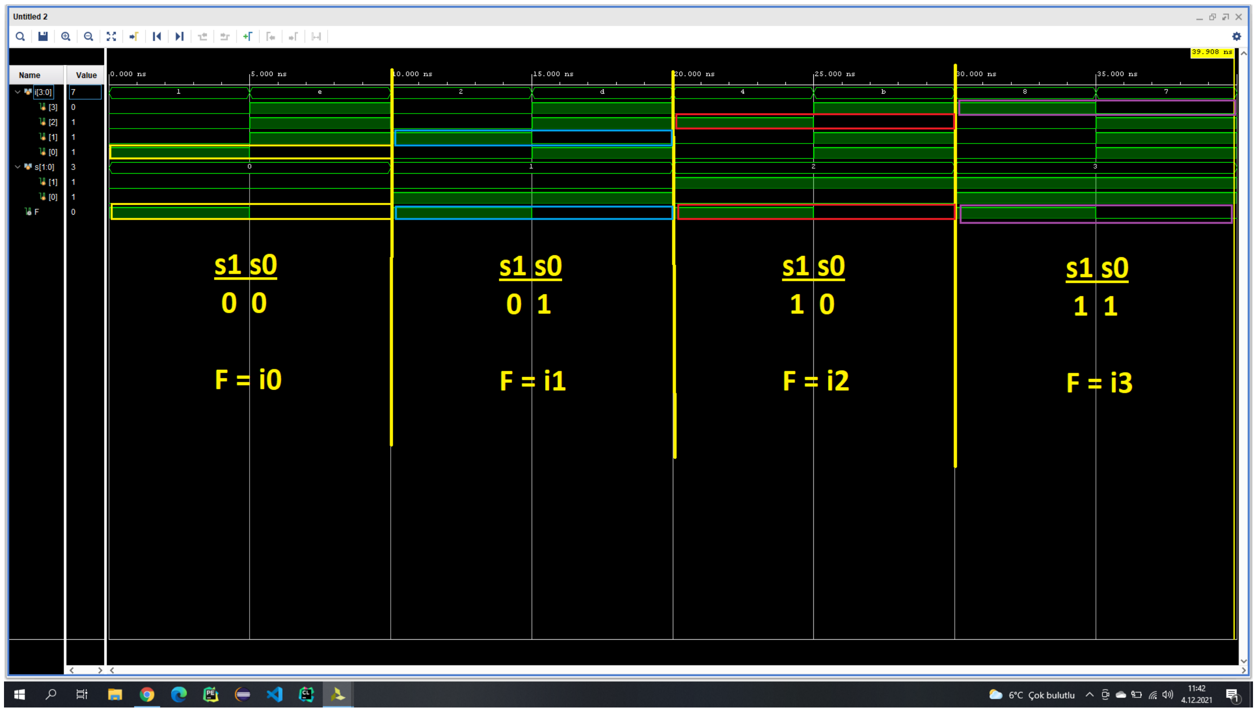

Figure 6: Waveform of the multiplexer explained

final circuit 7.

As you can see below, only the outputs (F's) in 0,5,10,15 are high. They are the minterms of our function.

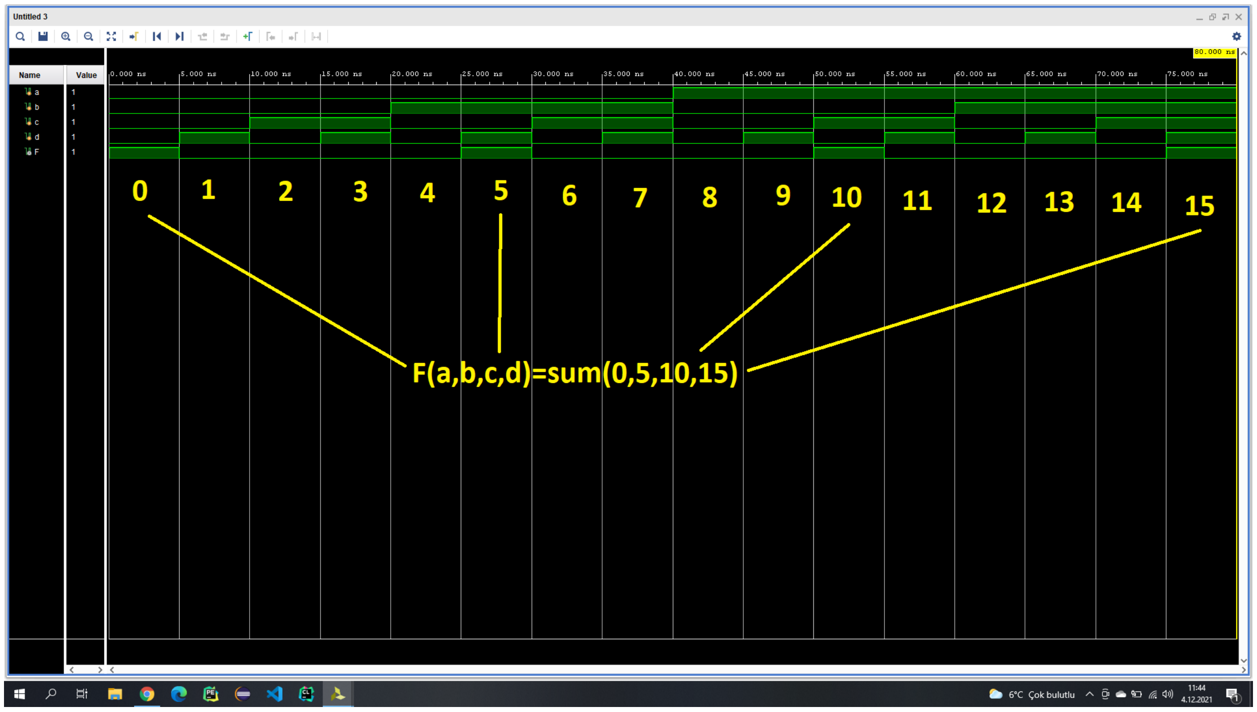

Figure 7: Waveform of the final circuit explained

5 Notes

In the circuit_tb.v, I have tried to test the circuit with a for loop in the beginning. But a had a problem with the every step in my code. Firstly, I have tried $\{a, b, c, d\} = i$ but since I declared "i" as an integer, waveforms did not operate properly. Then, I tried to declare i as a four bit number like $i = 4'b0000$ but it also did not work. In the end, I ran out of patience and wrote every test case manually. In my further projects, I will solve this problem and use loops for my test cases.

References

- BBM231 Lecture Notes
- BBM233 Lecture Notes