

DLD LAB WIN 2021LAB-1

Aim: To simulate different types of logic gates (NOT, NAND, NOR, XOR, XNOR) in different ways of modelling (DATAFLOW, BEHAVIORAL, STRUCTURAL)

Tools used: vivado software

Truth tables:

Truth tables :-

Not gate: $Y = \sim A$

A	Y(A complement)
0	1
1	0

Nand gate:-

$$Y = \sim(A \& B)$$

A	B	Y(A and B)
0	0	1
0	1	1
1	0	1
1	1	0

NOR gate: $Y = \sim(A + B)$

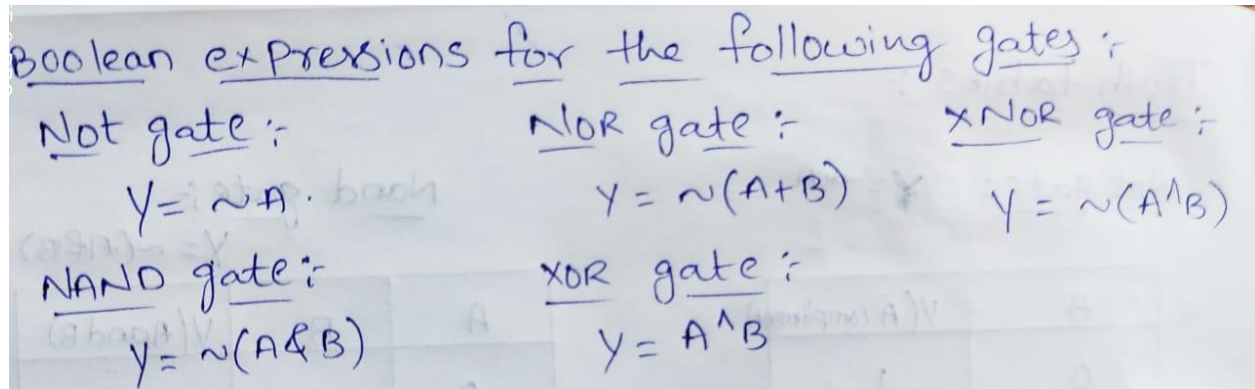
A	B	Y(A or B)
0	0	1
0	1	0
1	0	0
1	1	0

XOR gate:- $Y = A \wedge B$

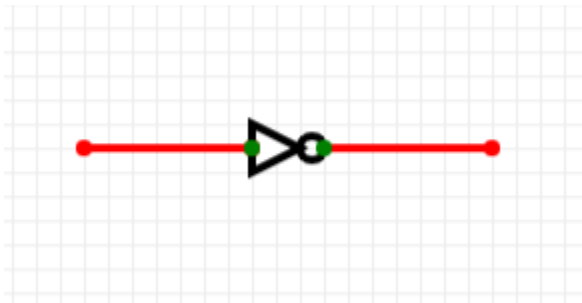
A	B	Y(A xor B)
0	0	0
0	1	1
1	0	1
1	1	0

XNOR gate:
 $Y = \sim(A \wedge B)$

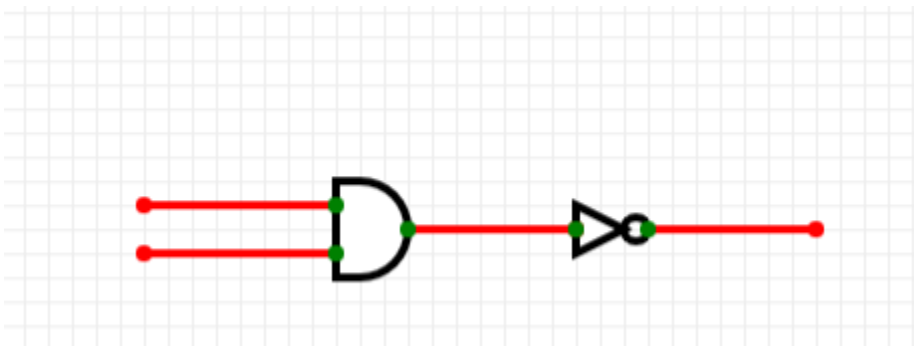
A	B	Y(A XNOR B)
0	0	1
0	1	0
1	0	0
1	1	1

Boolean expressions:**Circuit diagrams:**

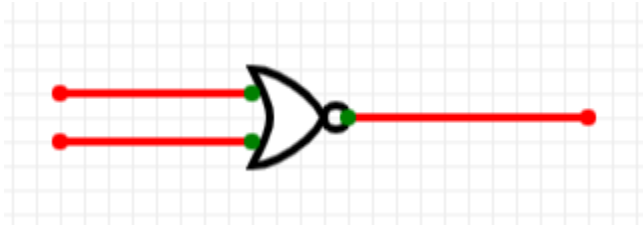
Not gate:



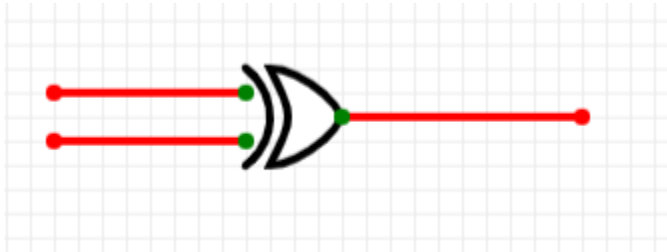
Nand gate:



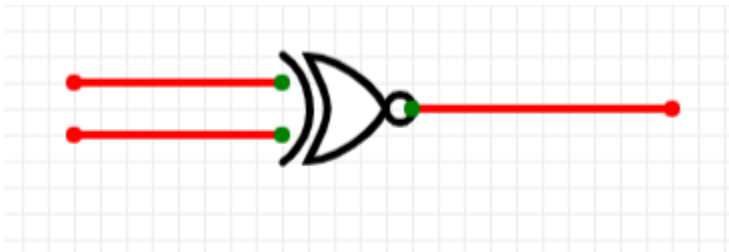
Nor gate:



Xor gate:

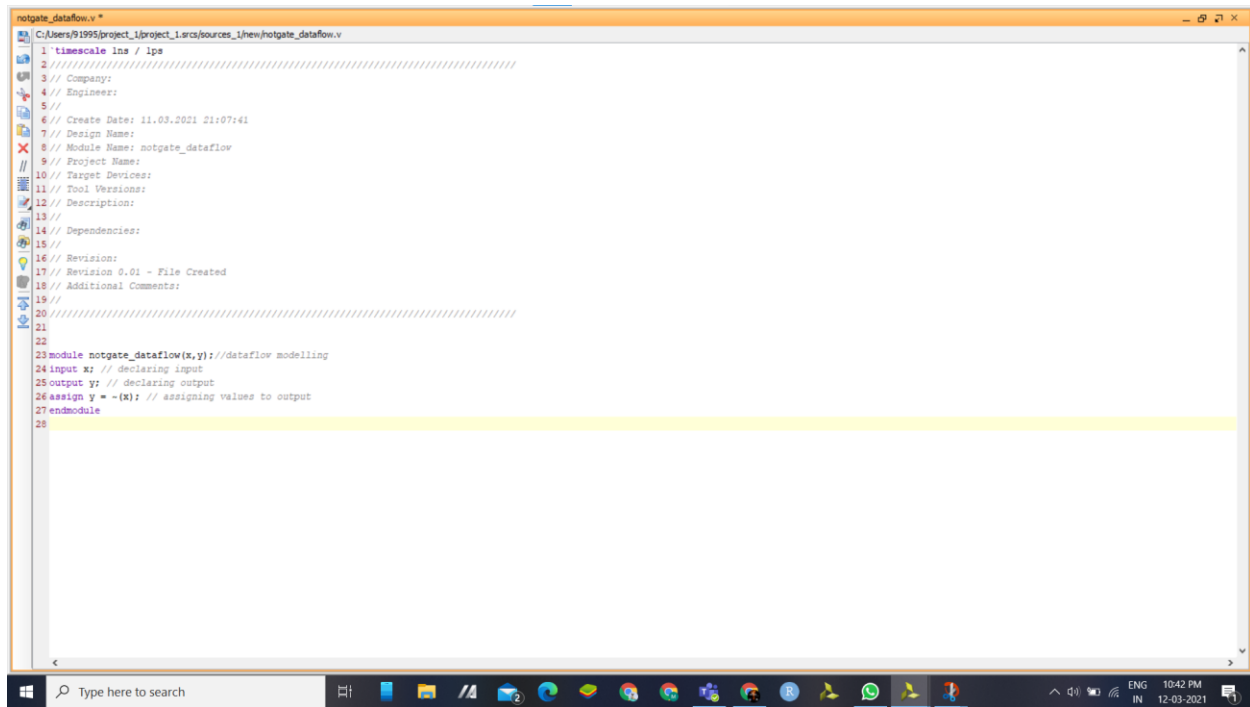


Xnor gate:



Codes:

Not gate:

A screenshot of a Verilog code editor window titled 'notgate_dataflow.v'. The code is as follows:

```
1 timescale 1ns / 1ps
2 //////////////////////////////////////////////////
3 // Company:
4 // Engineer:
5 //
6 // Create Date: 11.03.2021 21:07:41
7 // Design Name:
8 // Module Name: notgate_dataflow
9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revisions:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////
21
22
23 module notgate_dataflow(x,y)//dataflow modelling
24 input x; // declaring input
25 output y; // declaring output
26 assign y = ~(x); // assigning values to output
27 endmodule
28
```

The code is displayed in a text editor with a light blue background and a sidebar on the left showing file icons. The Windows taskbar is visible at the bottom.

module notgate_dataflow(x,y);//dataflow modelling

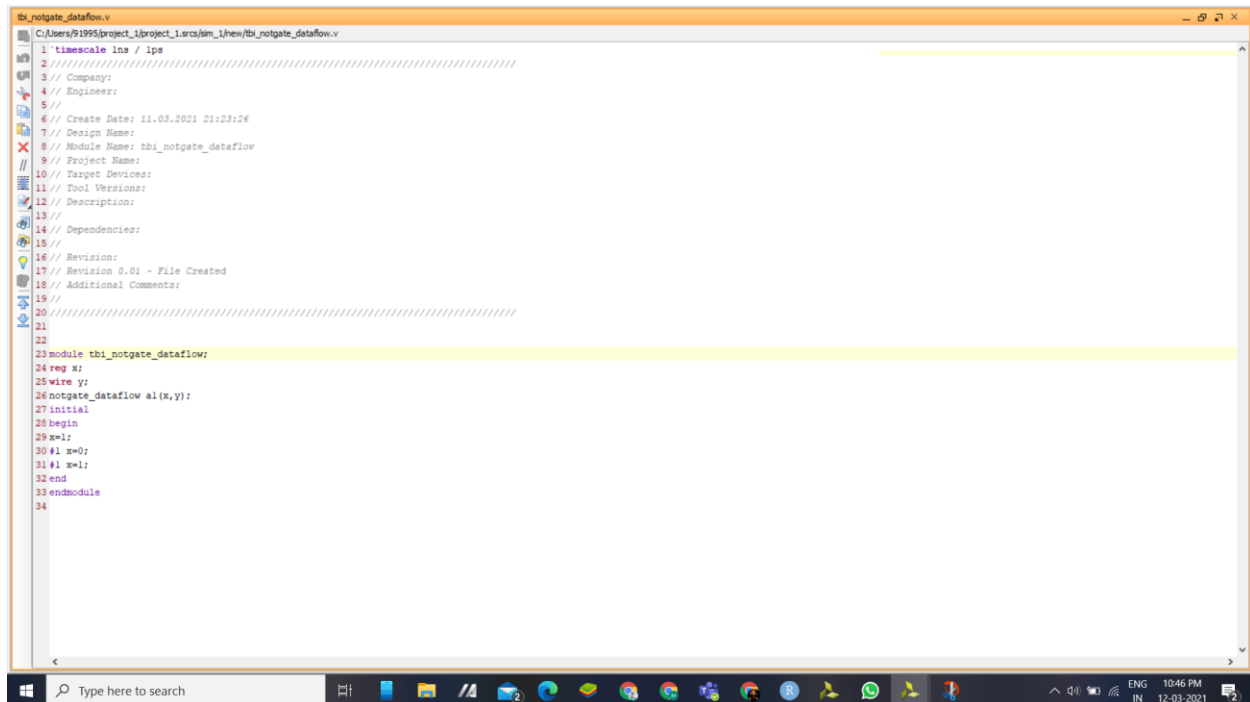
input x; // declaring input

output y; // declaring output

assign y = ~(x); // assigning values to output

Endmodule

Test bench code:

A screenshot of a text editor window titled 'tb_notgate_dataflow.v'. The editor shows a Verilog test bench code. The code includes a header section with comments for file name, company, engineer, date, design name, module name, project name, target devices, tool versions, description, dependencies, revision, and additional comments. The main code defines a module 'tbi_notgate_dataflow' with a register 'x', a wire 'y', and an instance of 'notgate_dataflow' named 'a1'. It includes an 'initial' block and a 'begin' block with assignments for 'x' and 'y'.

```
module tbi_notgate_dataflow;
```

```
reg x;
```

```
wire y;
```

```
notgate_dataflow a1(x,y);
```

```
initial
```

```
begin
```

```
x=1;
```

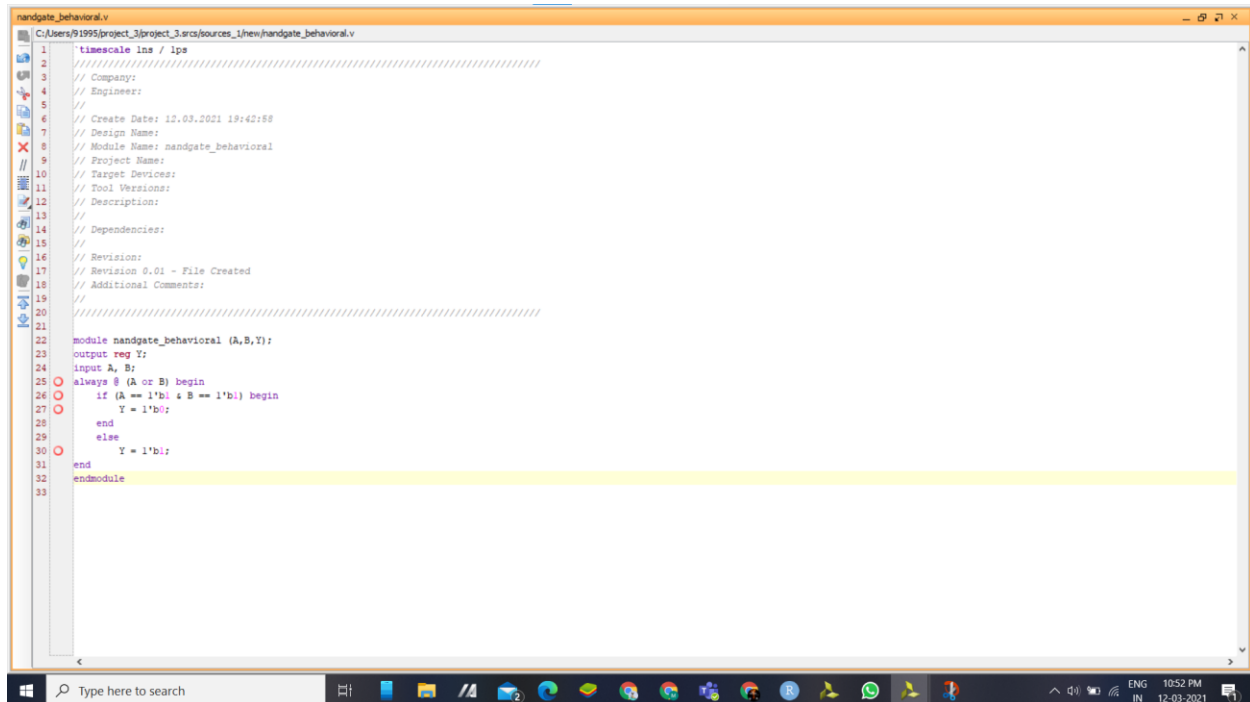
```
#1 x=0;
```

```
#1 x=1;
```

```
end
```

```
Endmodule
```

Nand gate:



```
nandgate_behavioral.v
C:\Users\91995\project_3\src\srcs\lnew\nandgate_behavioral.v
1 timescale 1ns / 1ps
2
3 // Company:
4 // Engineer:
5 //
6 // Create Date: 12.03.2021 19:42:58
7 // Design Name:
8 // Module Name: nandgate_behavioral
9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //
21
22 module nandgate_behavioral (A,B,Y);
23     output reg Y;
24     input A, B;
25     always @ (A or B) begin
26         if (A == 1'b1 & B == 1'b1) begin
27             Y = 1'b0;
28         end
29         else
30             Y = 1'b1;
31     end
32 endmodule
33
```

module nandgate_behavioral (A,B,Y);

output reg Y;

input A, B;

always @ (A or B) begin

if (A == 1'b1 & B == 1'b1) begin

Y = 1'b0;

end

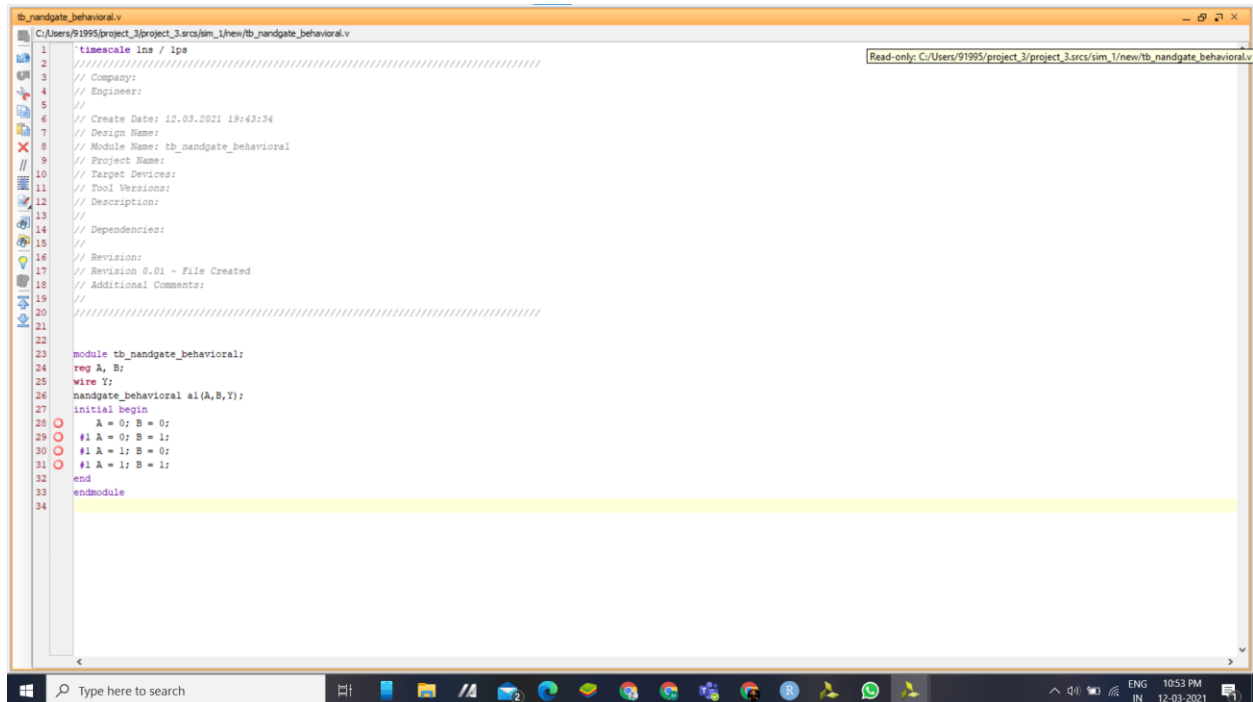
else

Y = 1'b1;

end

endmodule

Test bench code:

A screenshot of a text editor window showing Verilog test bench code. The code is for a module named 'tb_nandgate_behavioral'. It includes a header section with metadata like 'timescale', 'Company', 'Create Date', 'Design Name', 'Module Name', 'Project Name', 'Target Devices', 'Tool Versions', 'Description', 'Dependencies', 'Revision', and 'Additional Comments'. The main code block defines a module 'tb_nandgate_behavioral' with two registers 'A' and 'B', a wire 'Y', and an instance of a 'nandgate_behavioral' module 'a1'. An 'initial' block contains four test cases for the nand gate: (0,0), (0,1), (1,0), and (1,1). The code is written in a text editor with a light blue background and a sidebar on the left showing file explorer icons. The Windows taskbar is visible at the bottom.

```
1 timescale 1ns / 1ps
2
3 // Company:
4 // Engineer:
5 //
6 // Create Date: 12.03.2021 19:43:34
7 // Design Name:
8 // Module Name: tb_nandgate_behavioral
9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //
21
22
23 module tb_nandgate_behavioral:
24 reg A, B;
25 wire Y;
26 nandgate_behavioral a1(A,B,Y);
27 initial begin
28     A = 0; B = 0;
29     #1 A = 0; B = 1;
30     #1 A = 1; B = 0;
31     #1 A = 1; B = 1;
32 end
33 endmodule
34
```

```
module tb_nandgate_behavioral;
```

```
reg A, B;
```

```
wire Y;
```

```
nandgate_behavioral a1(A,B,Y);
```

```
initial begin
```

```
    A = 0; B = 0;
```

```
    #1 A = 0; B = 1;
```

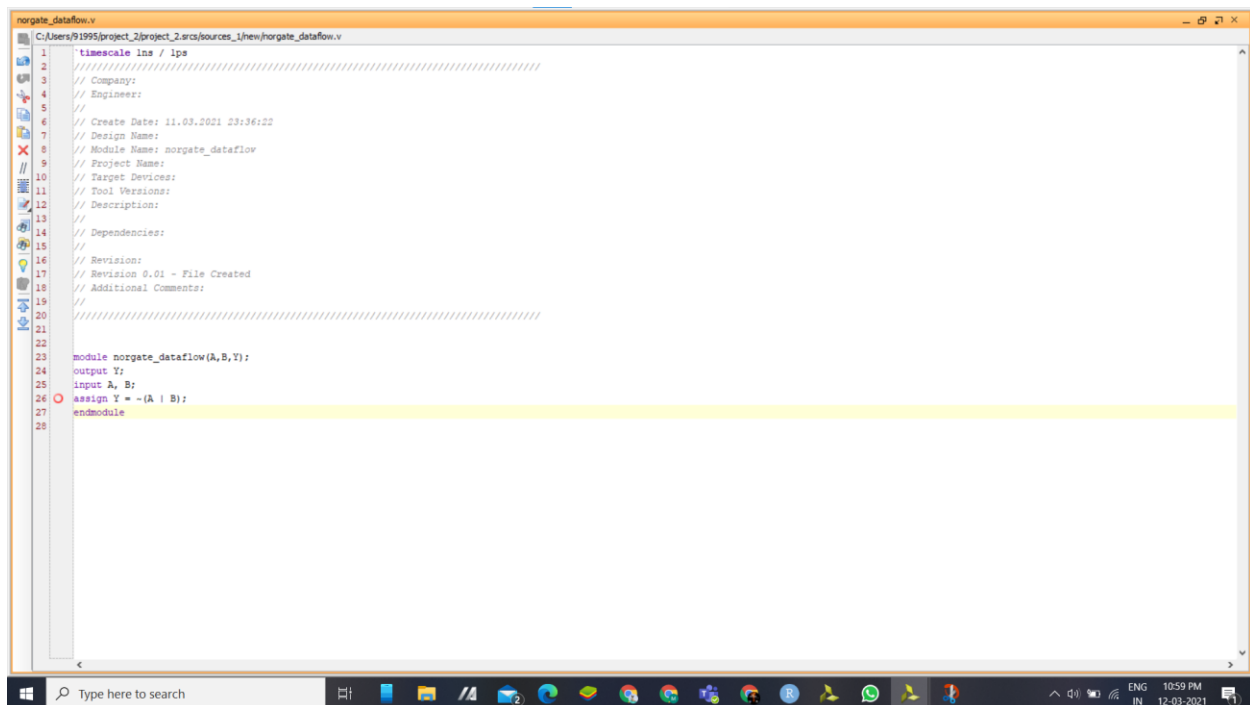
```
    #1 A = 1; B = 0;
```

```
    #1 A = 1; B = 1;
```

```
end
```

endmodule

Nor gate:



The screenshot shows a Verilog code editor window titled 'norgate_dataflow.v'. The code is as follows:

```
1 timescale 1ns / 1ps
2 //////////////////////////////////////////////////
3 // Company:
4 // Engineer:
5 //
6 // Create Date: 11.03.2021 23:36:22
7 // Design Name:
8 // Module Name: norgate_dataflow
9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////
21
22 module norgate_dataflow(A,B,Y);
23   output Y;
24   input A, B;
25   assign Y = ~(A | B);
26 endmodule
27
28
```

module norgate_dataflow(A,B,Y);

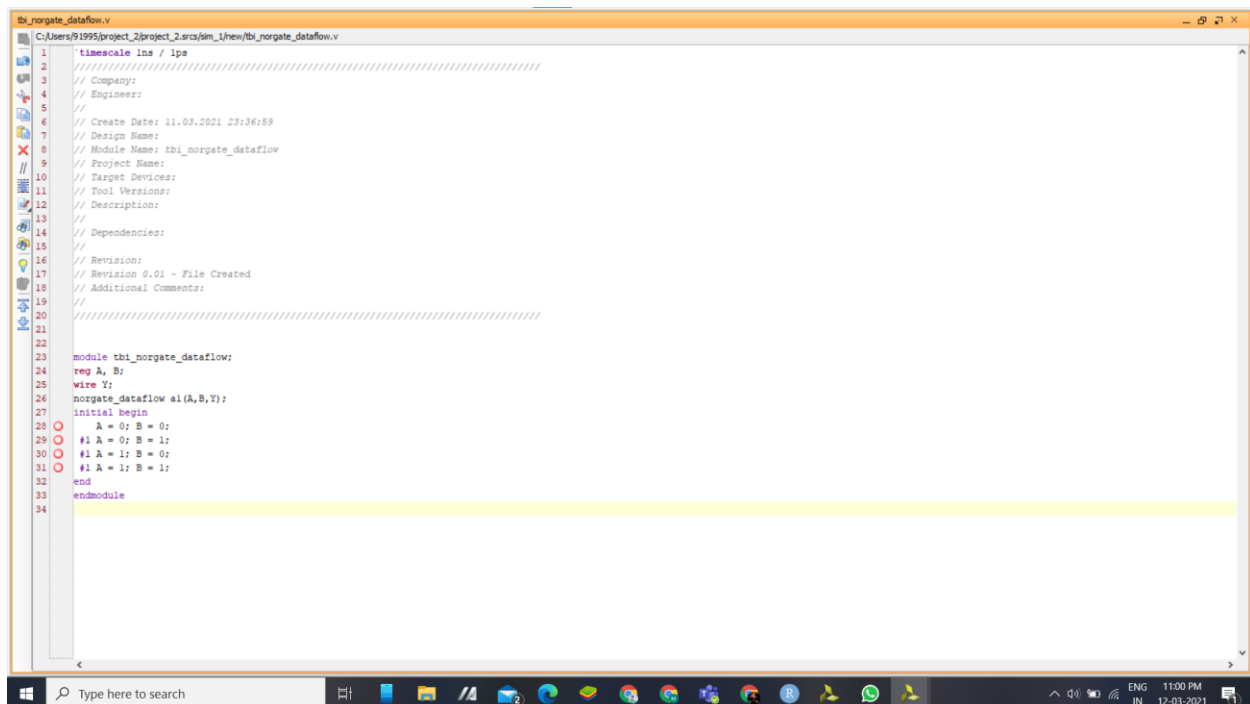
output Y;

input A, B;

assign Y = ~(A | B);

endmodule

Test bench code:

A screenshot of a text editor window showing Verilog test bench code. The code is for a module named 'tbi_norgate_dataflow'. It includes a header section with metadata like 'timescale', 'Company', 'Engineer', 'Create Date', 'Design Name', 'Module Name', 'Project Name', 'Target Devices', 'Tool Versions', 'Description', 'Dependencies', 'Revision', and 'Additional Comments'. The main code defines a module 'tbi_norgate_dataflow' with two inputs 'A' and 'B', and one output 'Y'. It instantiates a component 'norgate_dataflow a1(A,B,Y)'. The test bench logic is enclosed in an 'initial begin' block, which sets 'A = 0; B = 0;' and then applies four test cases: '#1 A = 0; B = 1;', '#1 A = 1; B = 0;', and '#1 A = 1; B = 1;'. The code ends with 'end' and 'endmodule'. The editor has a yellow highlight on the line containing 'endmodule'. The Windows taskbar is visible at the bottom with the date and time '12-03-2021 11:00 PM'.

```
module tbi_norgate_dataflow;
```

```
reg A, B;
```

```
wire Y;
```

```
norgate_dataflow a1(A,B,Y);
```

```
initial begin
```

```
    A = 0; B = 0;
```

```
    #1 A = 0; B = 1;
```

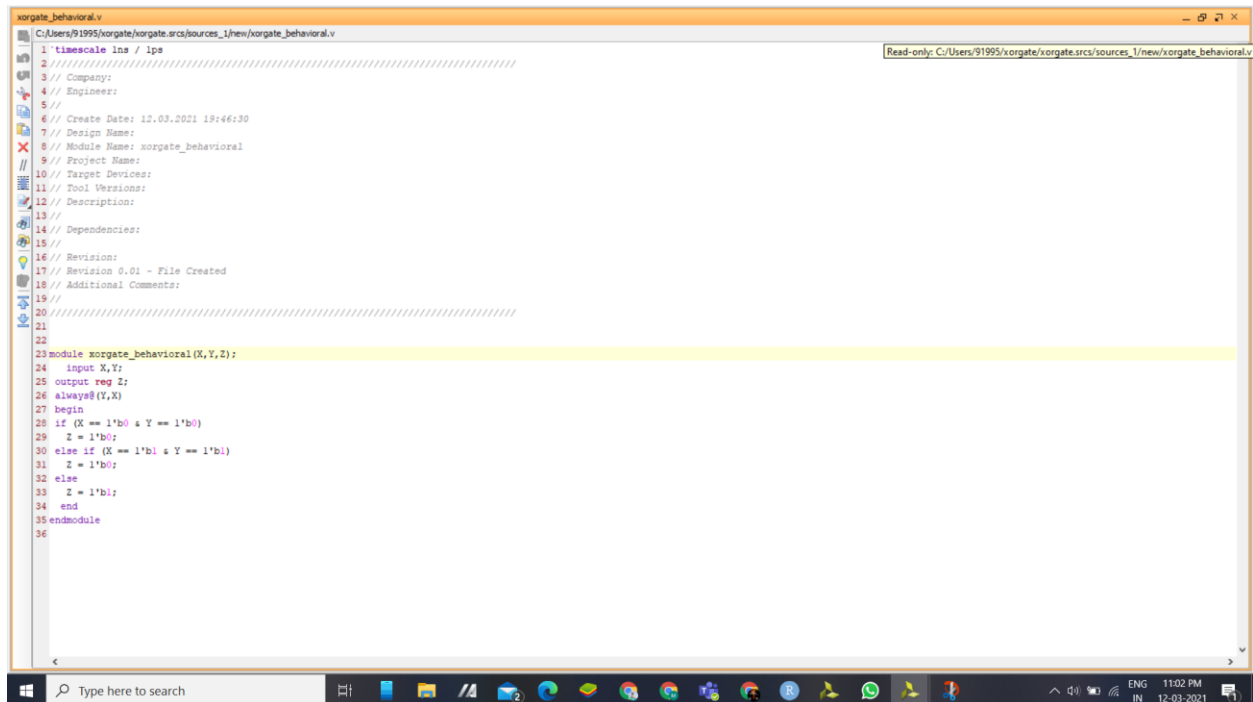
```
    #1 A = 1; B = 0;
```

```
    #1 A = 1; B = 1;
```

```
end
```

endmodule

Xor gate:



The screenshot shows a Verilog code editor window titled 'xorgate_behavioral.v'. The code defines a module 'xorgate_behavioral' with inputs 'X' and 'Y', and output 'Z'. It implements an XOR gate using a conditional statement. The code is as follows:

```
1 timescale 1ns / 1ps
2 //////////////////////////////////////////////////
3 // Company:
4 // Engineer:
5 //
6 // Create Date: 12.03.2021 19:46:30
7 // Design Name:
8 // Module Name: xorgate_behavioral
9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////
21
22
23 module xorgate_behavioral(X,Y,Z);
24     input X,Y;
25     output reg Z;
26     always@(X,Y)
27     begin
28         if (X == 1'b0 & Y == 1'b0)
29             Z = 1'b0;
30         else if (X == 1'b1 & Y == 1'b1)
31             Z = 1'b0;
32         else
33             Z = 1'b1;
34         end
35     endmodule
36
```

module xorgate_behavioral(X,Y,Z);

input X,Y;

output reg Z;

always@(Y,X)

begin

if (X == 1'b0 & Y == 1'b0)

Z = 1'b0;

else if (X == 1'b1 & Y == 1'b1)

Z = 1'b0;

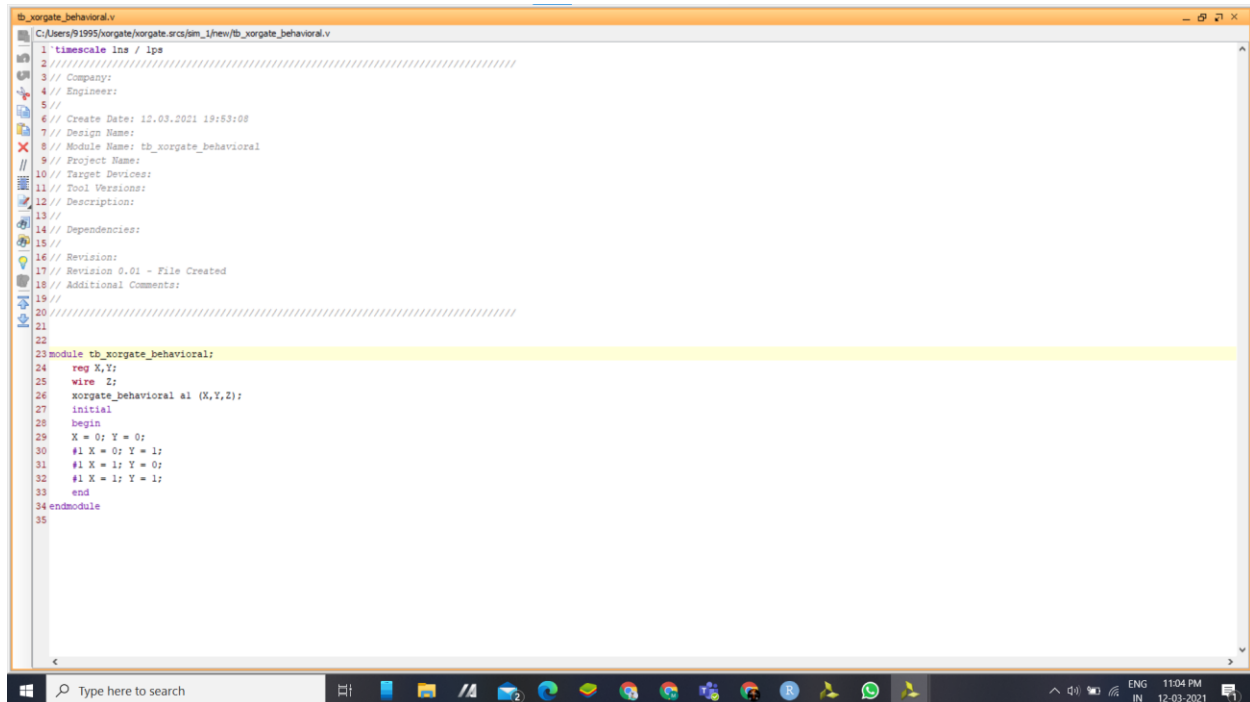
else

Z = 1'b1;

end

endmodule

Test bench code:

A screenshot of a text editor window titled 'tb_xorgate_behavioral.v'. The editor shows Verilog code for a test bench. The code includes a header section with comments for company, engineer, date, design name, module name, project name, target devices, tool versions, description, dependencies, revision, and additional comments. The main code defines a module 'tb_xorgate_behavioral' with three registers: 'X', 'Y', and 'Z'. It instantiates the 'xorgate_behavioral' module as 'a1' with inputs 'X', 'Y', and 'Z'. The test bench includes an 'initial' block and a 'begin' block. The 'initial' block sets 'X = 0; Y = 0;'. The 'begin' block contains four test cases: '#1 X = 0; Y = 1;', '#1 X = 1; Y = 0;', and '#1 X = 1; Y = 1;'. The code ends with 'end' and 'endmodule'.

```
module tb_xorgate_behavioral;
```

```
    reg X,Y;
```

```
    wire Z;
```

```
    xorgate_behavioral a1 (X,Y,Z);
```

```
    initial
```

```
    begin
```

```
        X = 0; Y = 0;
```

```
        #1 X = 0; Y = 1;
```

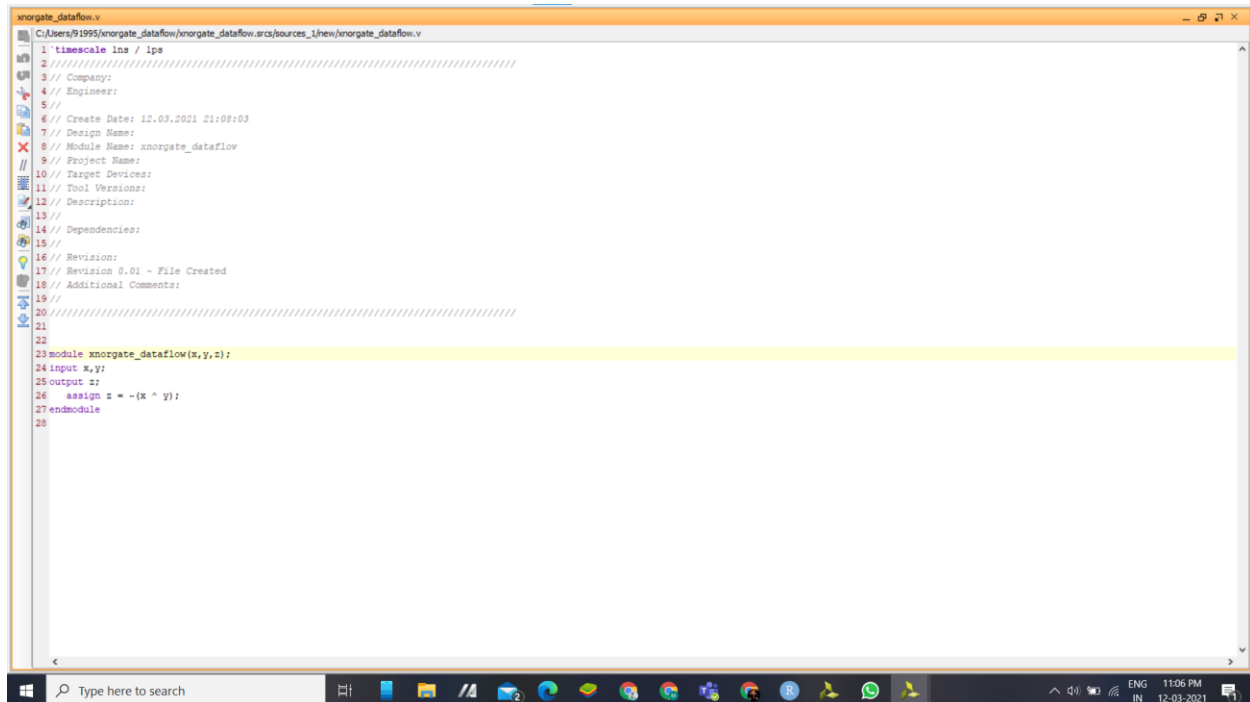
```
        #1 X = 1; Y = 0;
```

```
        #1 X = 1; Y = 1;
```

end

endmodule

xnor gate:



The screenshot shows a Verilog code editor window titled 'xnorgate_dataflow.v'. The code is as follows:

```
1 timescale 1ns / 1ps
2 //////////////////////////////////////////////////
3 // Company:
4 // Engineer:
5 //
6 // Create Date: 12.03.2021 21:08:03
7 // Design Name:
8 // Module Name: xnorgate_dataflow
9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////
21
22
23 module xnorgate_dataflow(x,y,z);
24 input x,y;
25 output z;
26 assign z = ~(x ^ y);
27 endmodule
28
```

module xnorgate_dataflow(x,y,z);

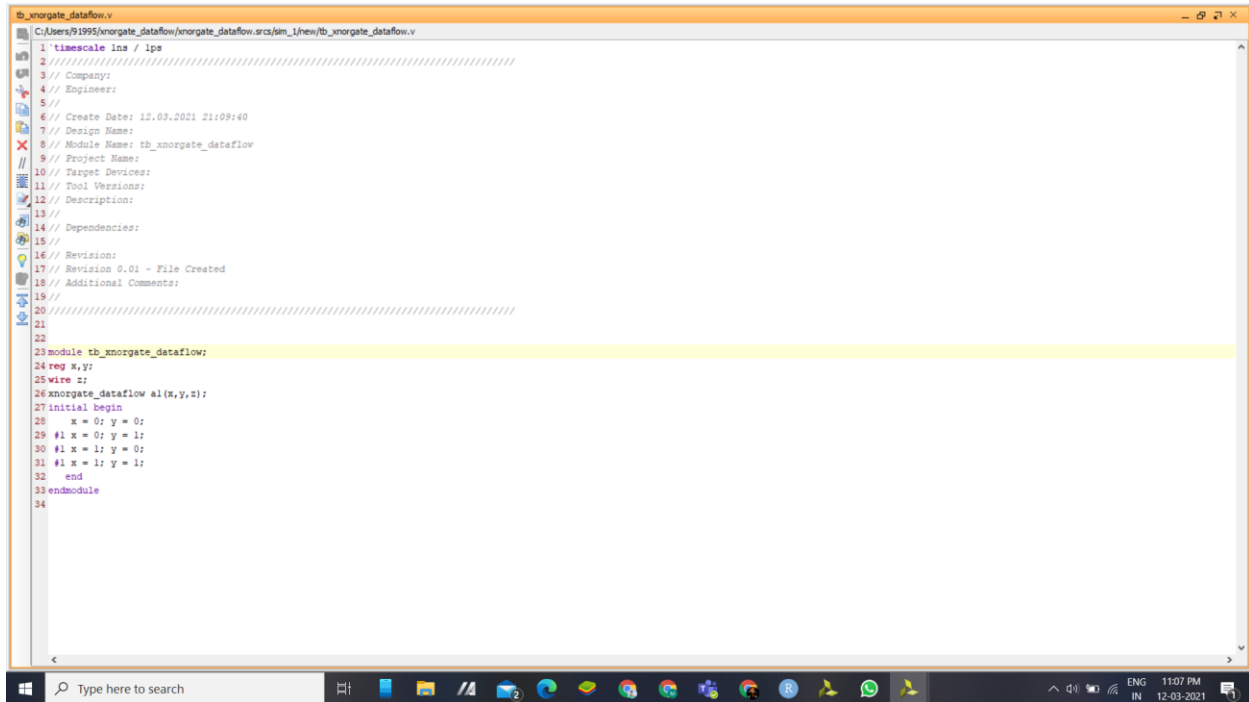
input x,y;

output z;

assign z = ~(x ^ y);

endmodule

Test bench code:

A screenshot of a text editor window titled 'tb_xnorgate_dataflow.v'. The editor shows a Verilog testbench code. The code starts with a comment block containing metadata like 'Create Date: 12.03.2021 21:09:40', 'Design Name: tb_xnorgate_dataflow', and 'Project Name:'. Below the comments, the code defines a module 'tb_xnorgate_dataflow' with two registers 'x' and 'y', and a wire 'z'. It then instantiates the 'xnorgate_dataflow' module 'a1' with inputs 'x', 'y', and 'z'. An 'initial' block sets the initial values of 'x' and 'y' to 0. Finally, the code includes a series of assertions to check the output 'z' for specific input combinations: (0,0), (0,1), (1,0), and (1,1). The code ends with 'endmodule' and 'end'.

```
module tb_xnorgate_dataflow;
```

```
reg x,y;
```

```
wire z;
```

```
xnorgate_dataflow a1(x,y,z);
```

```
initial begin
```

```
    x = 0; y = 0;
```

```
#1 x = 0; y = 1;
```

```
#1 x = 1; y = 0;
```

```
#1 x = 1; y = 1;
```

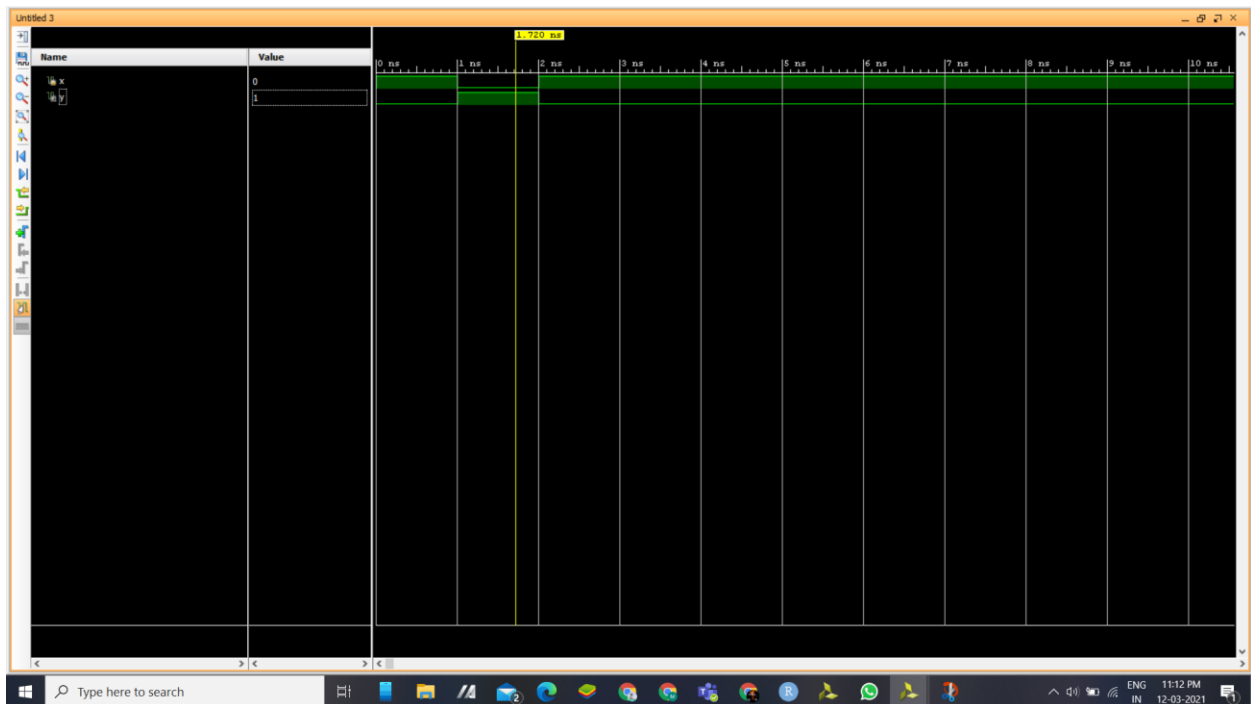
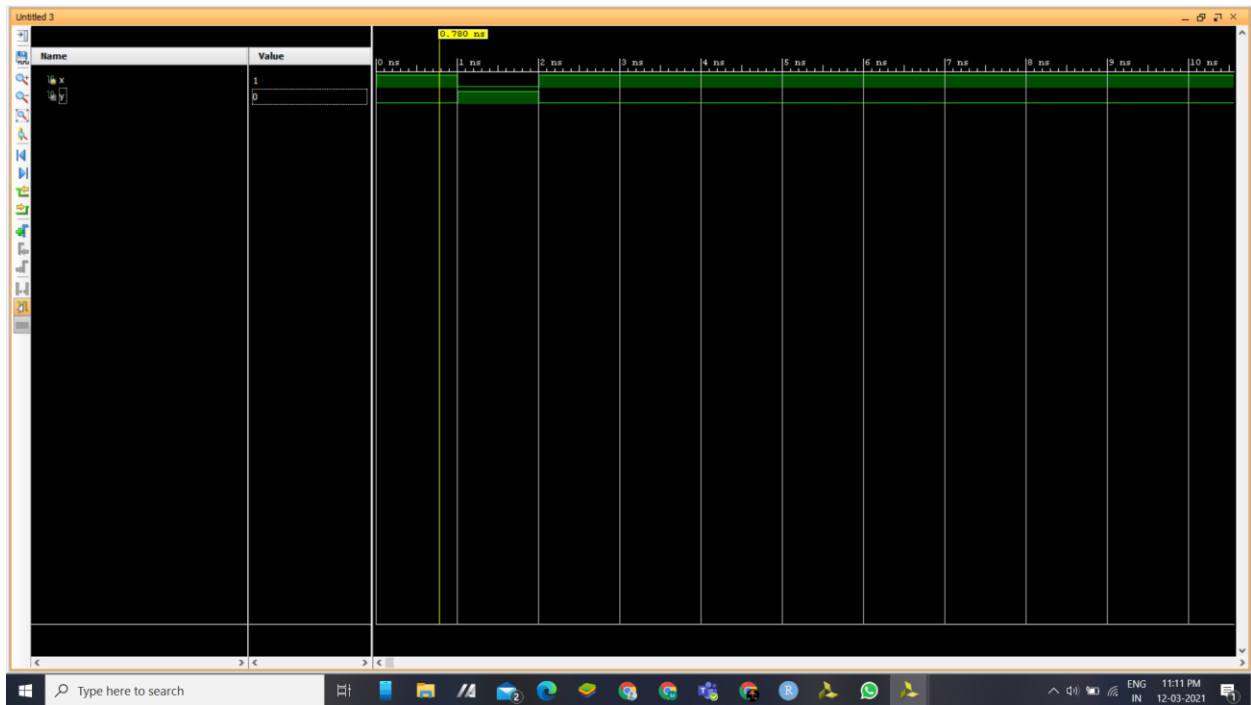
```
    end
```

```
endmodule
```

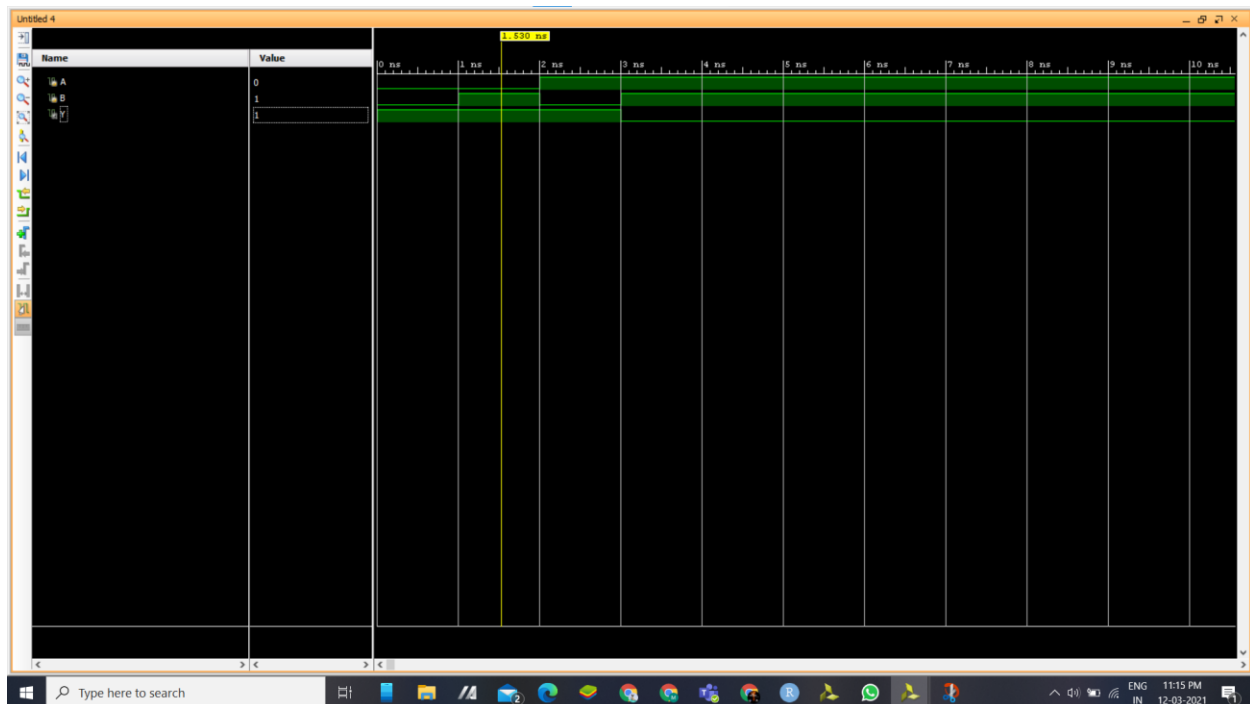
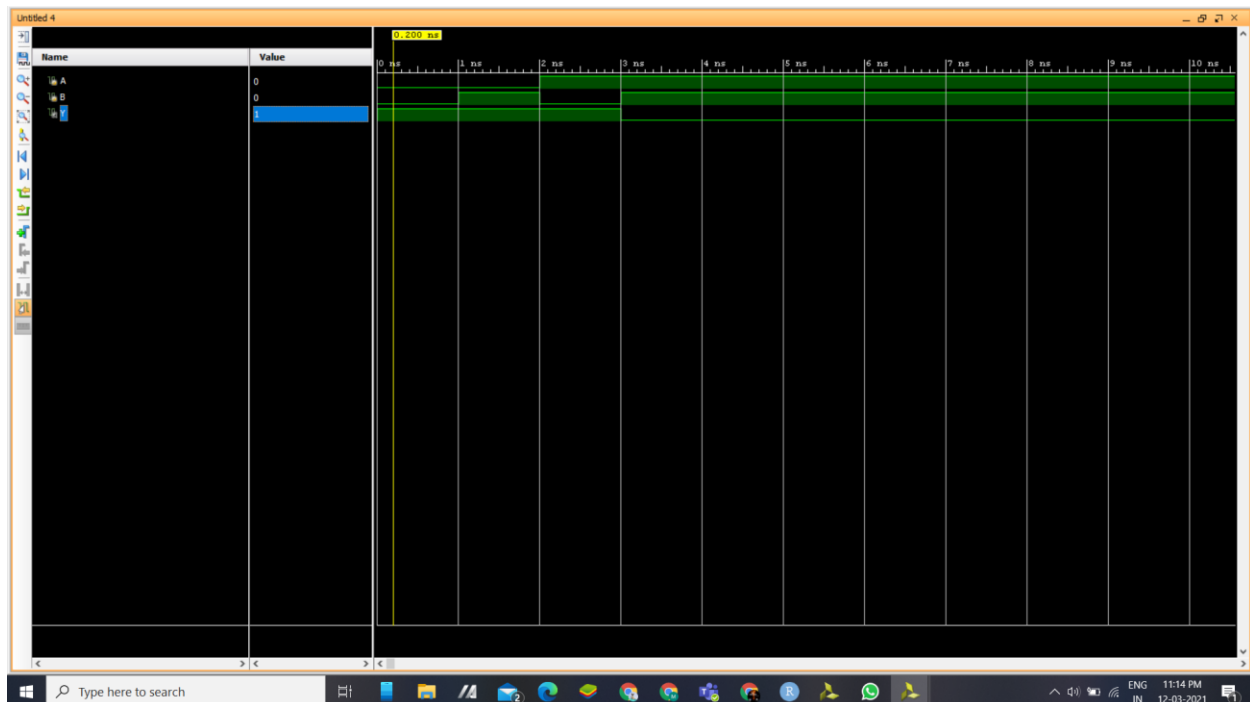
Results of codes:

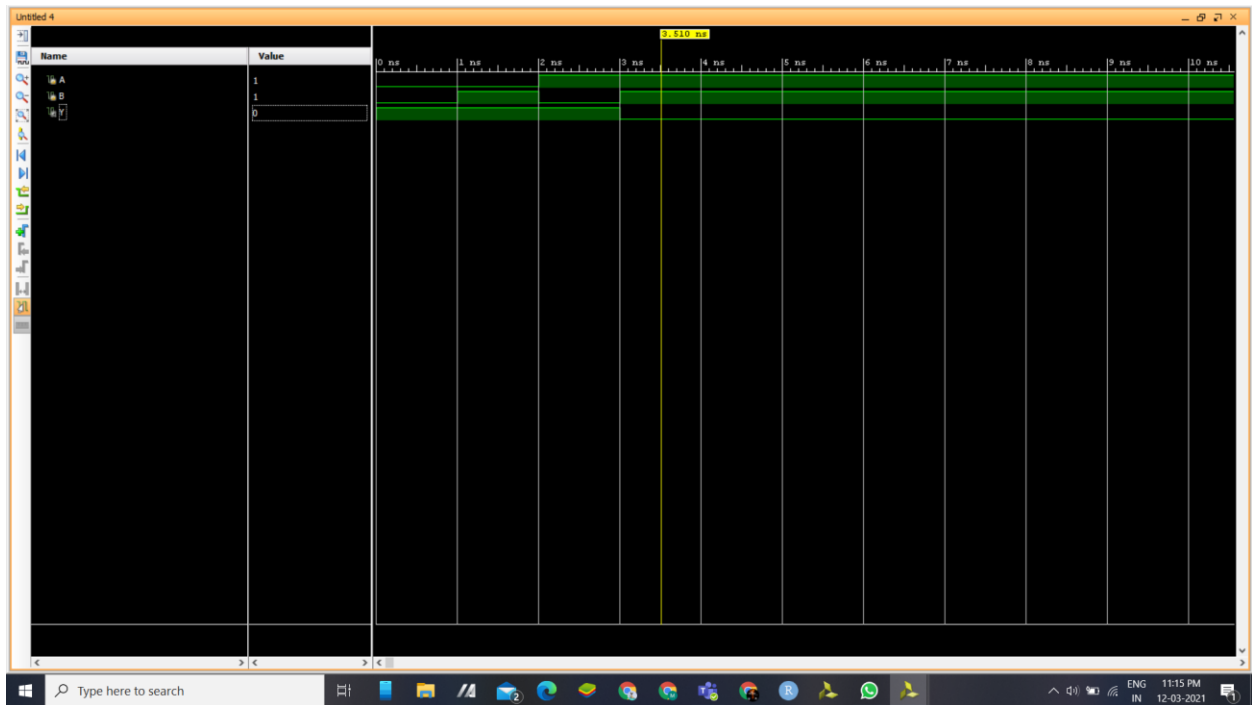
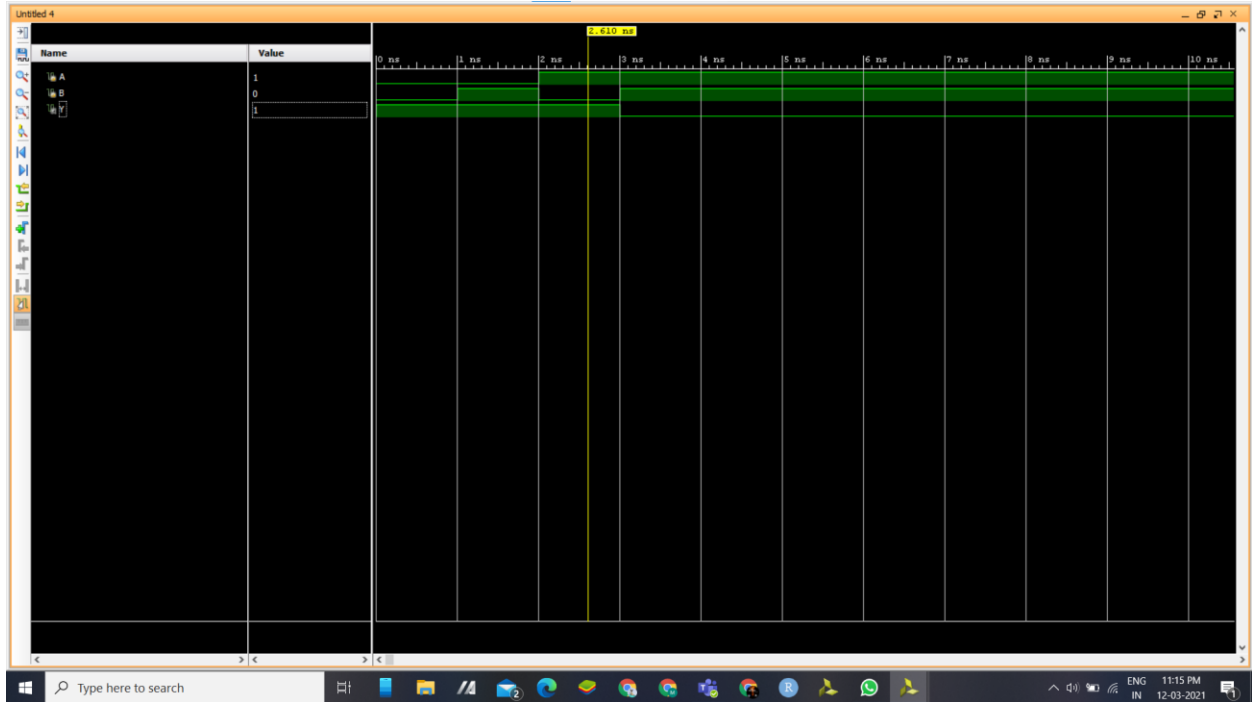
>>>>><<<<<

Not gate:

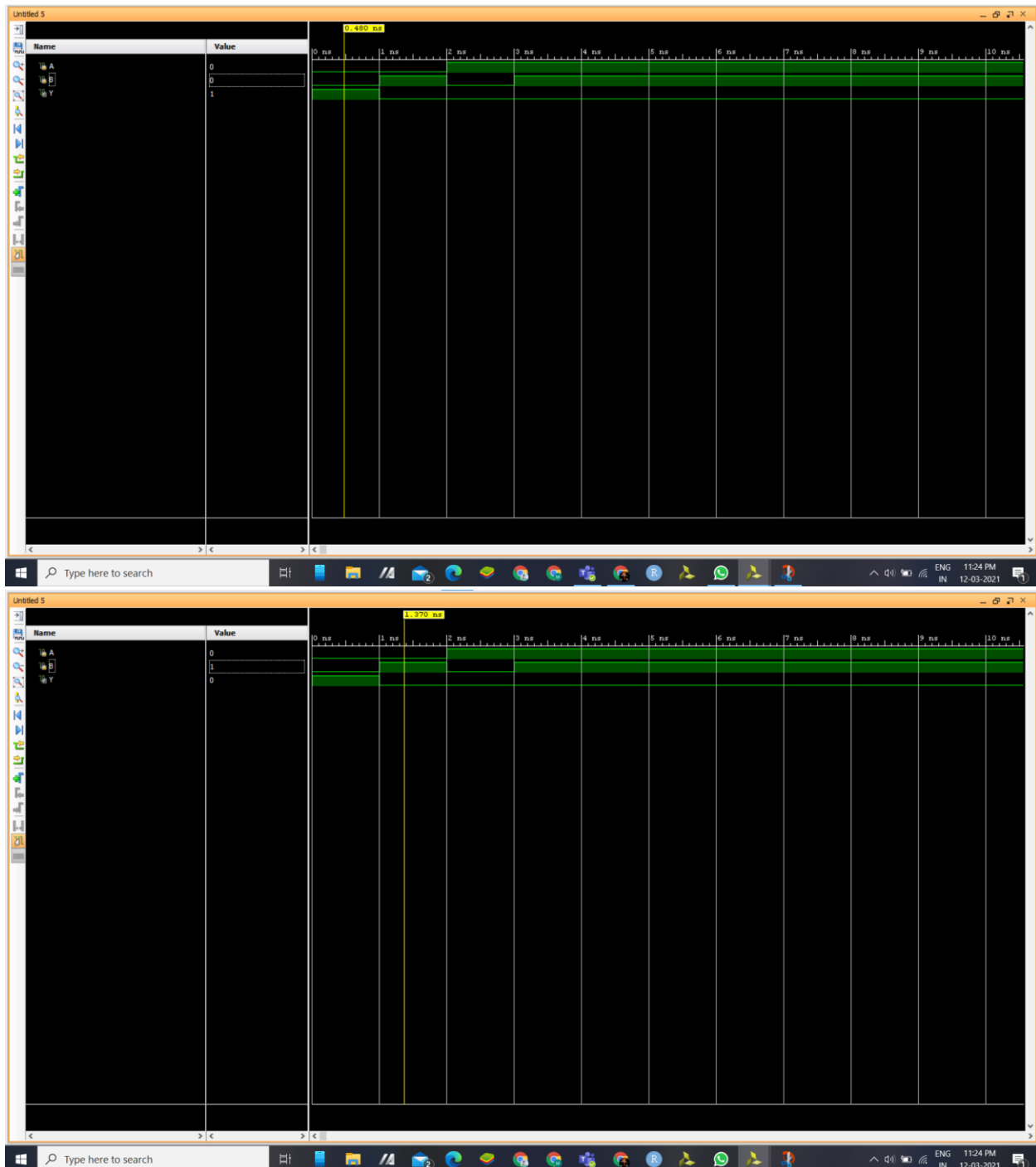


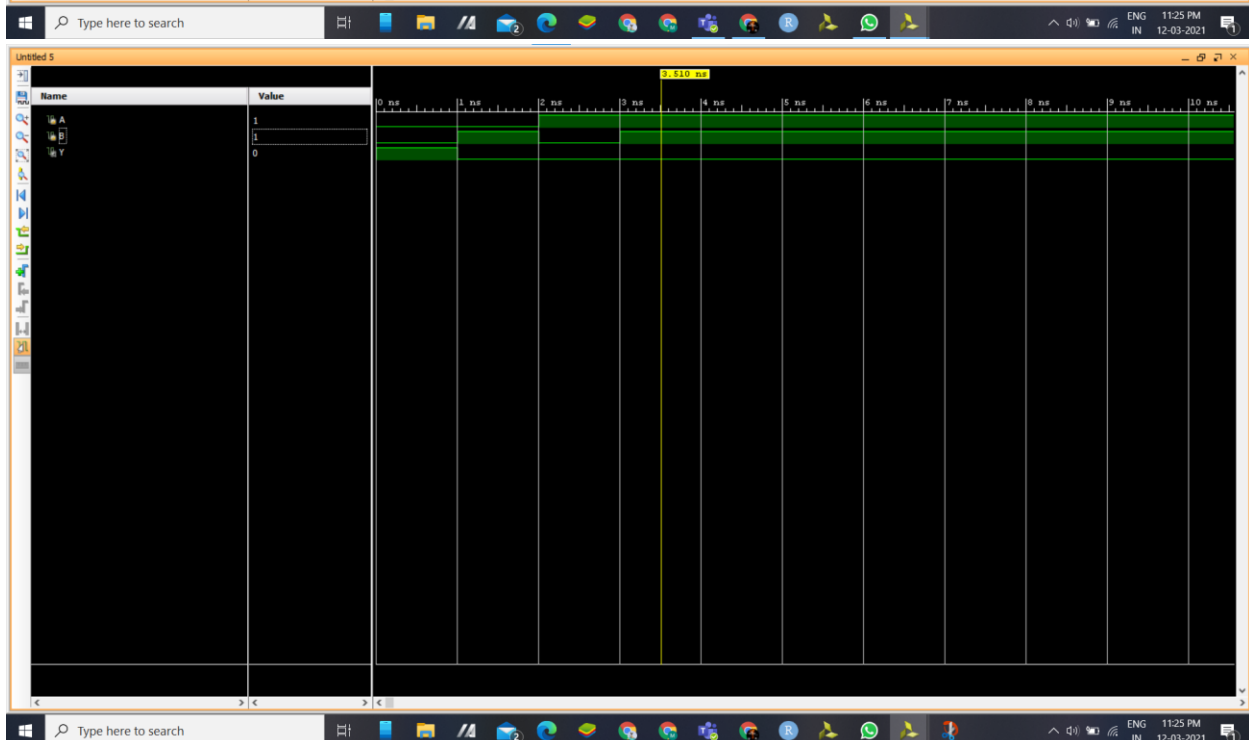
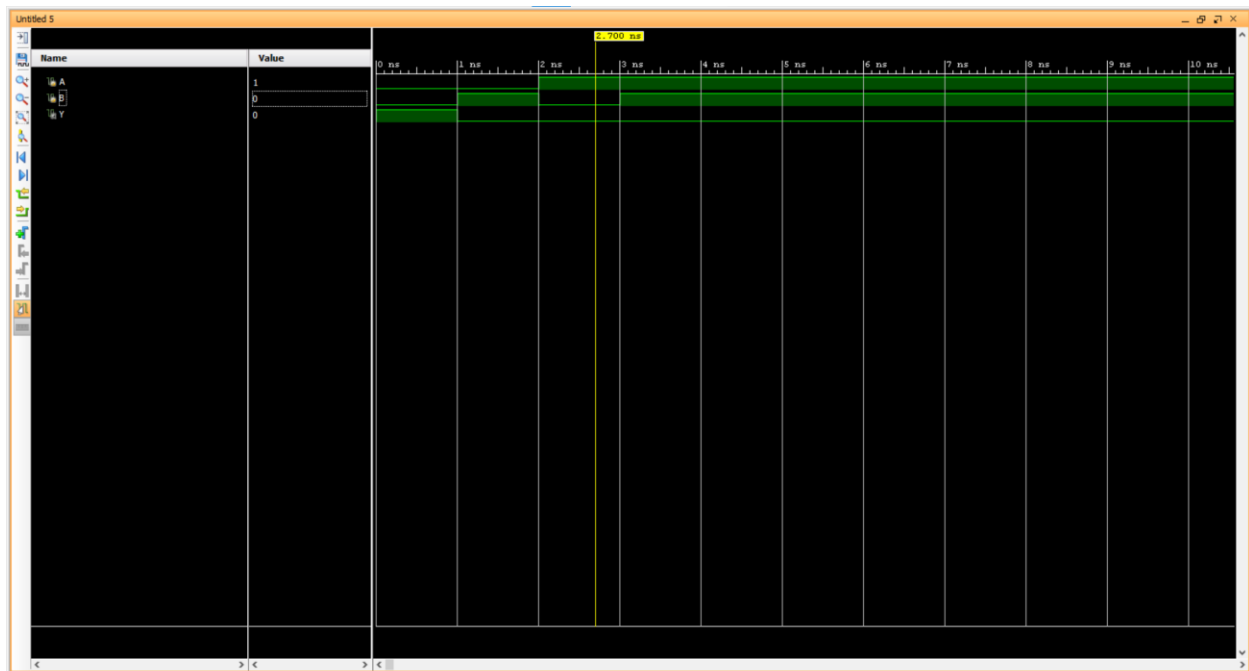
Nand gate:



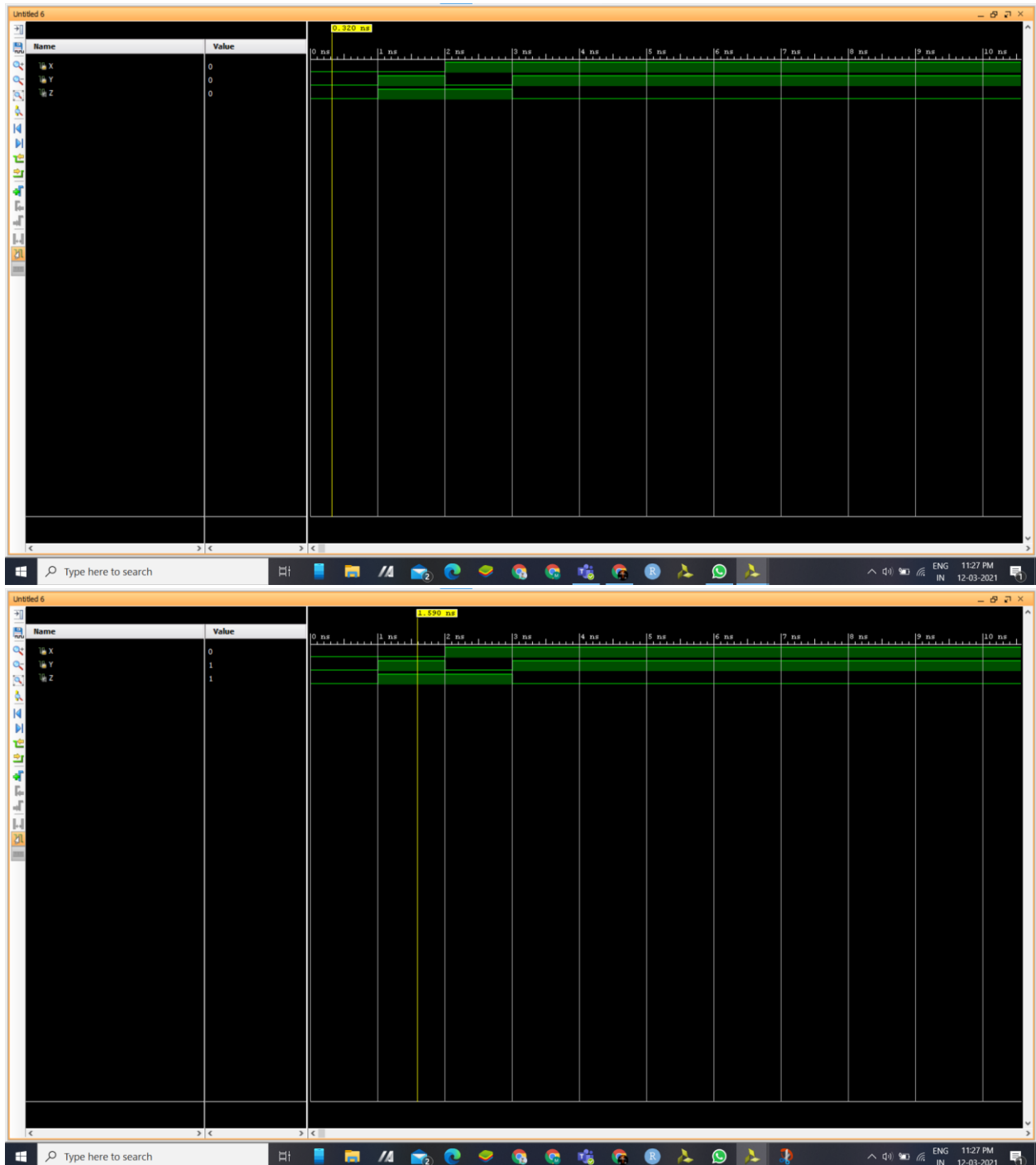


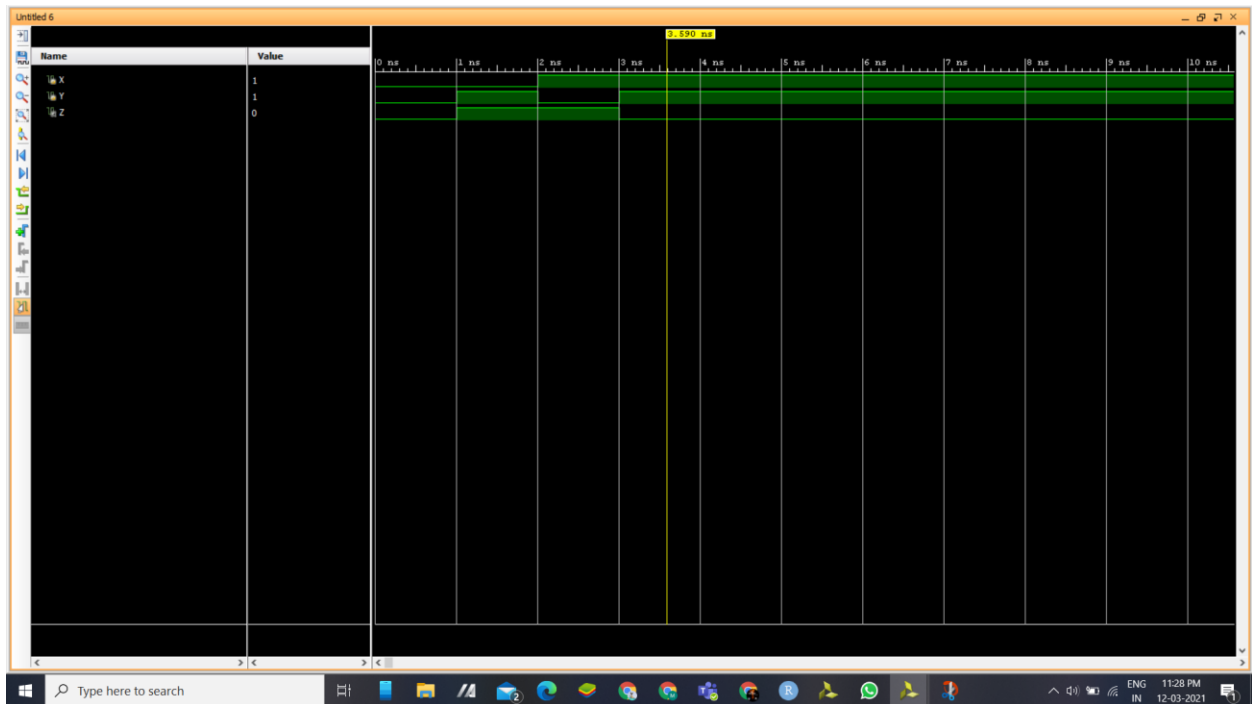
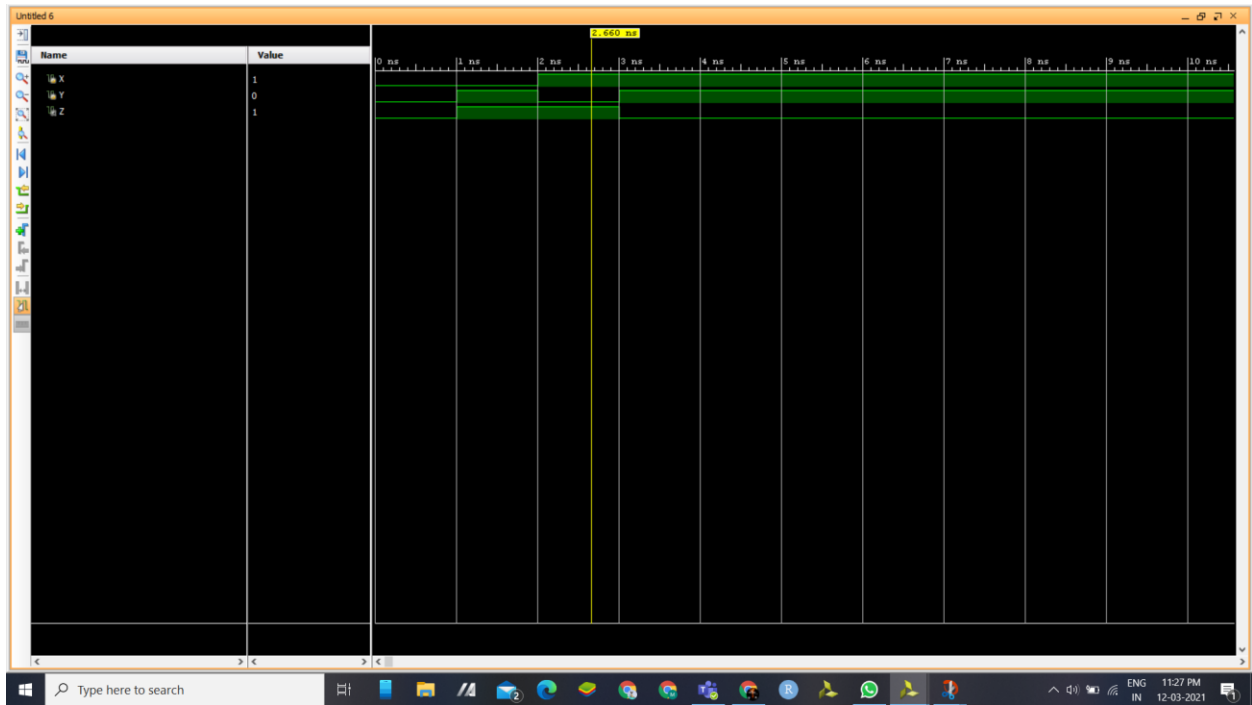
Nor gate:



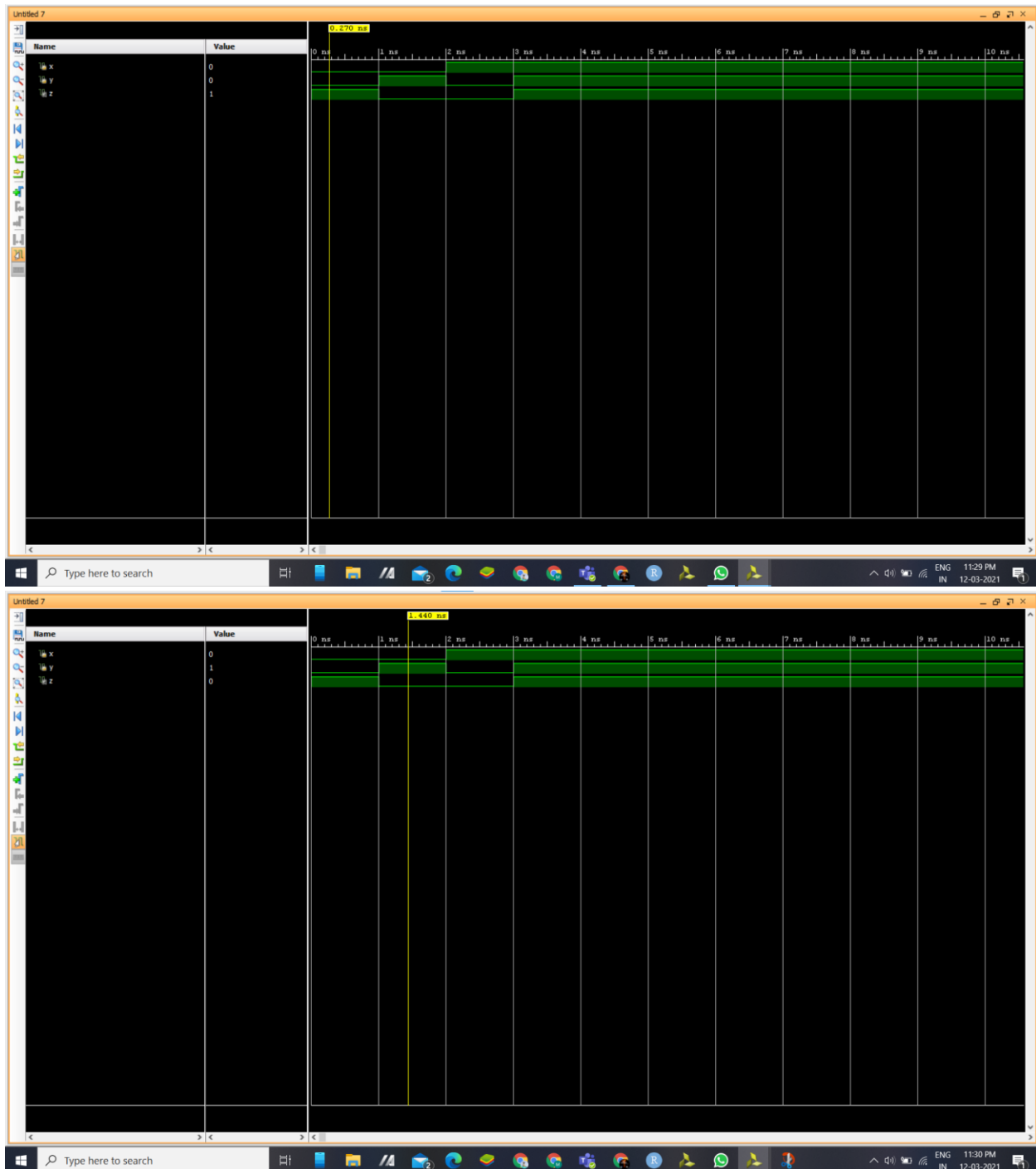


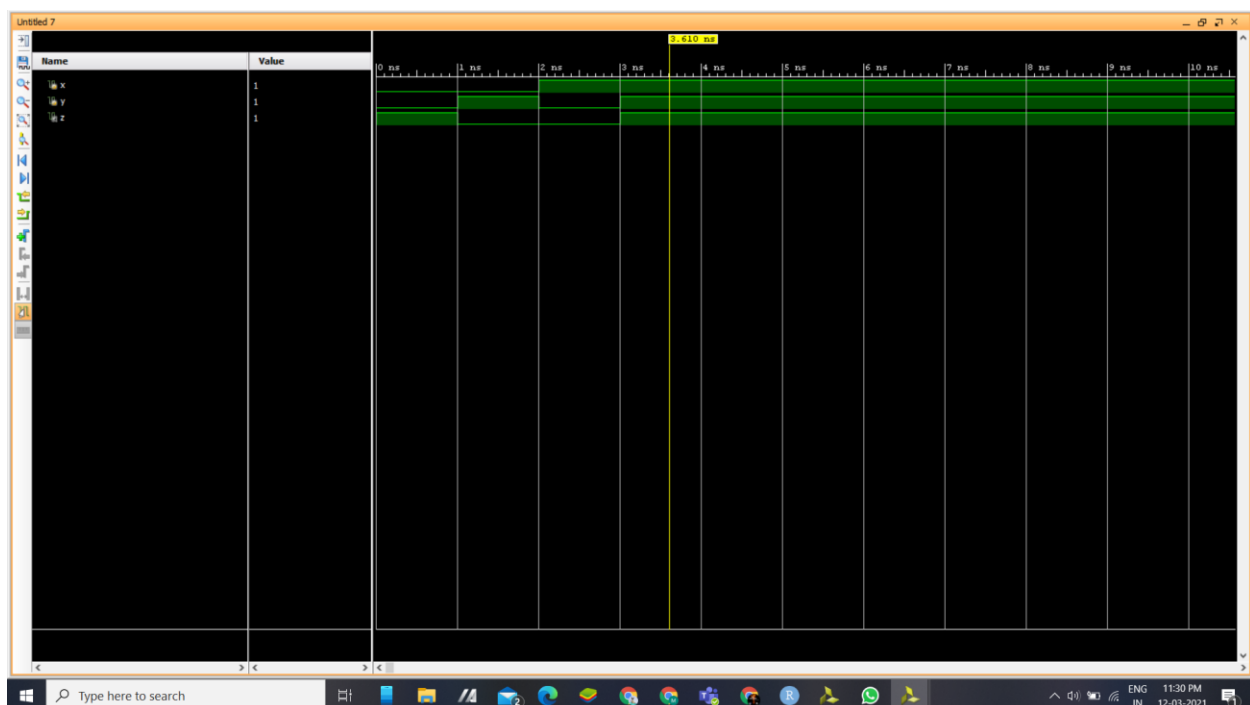
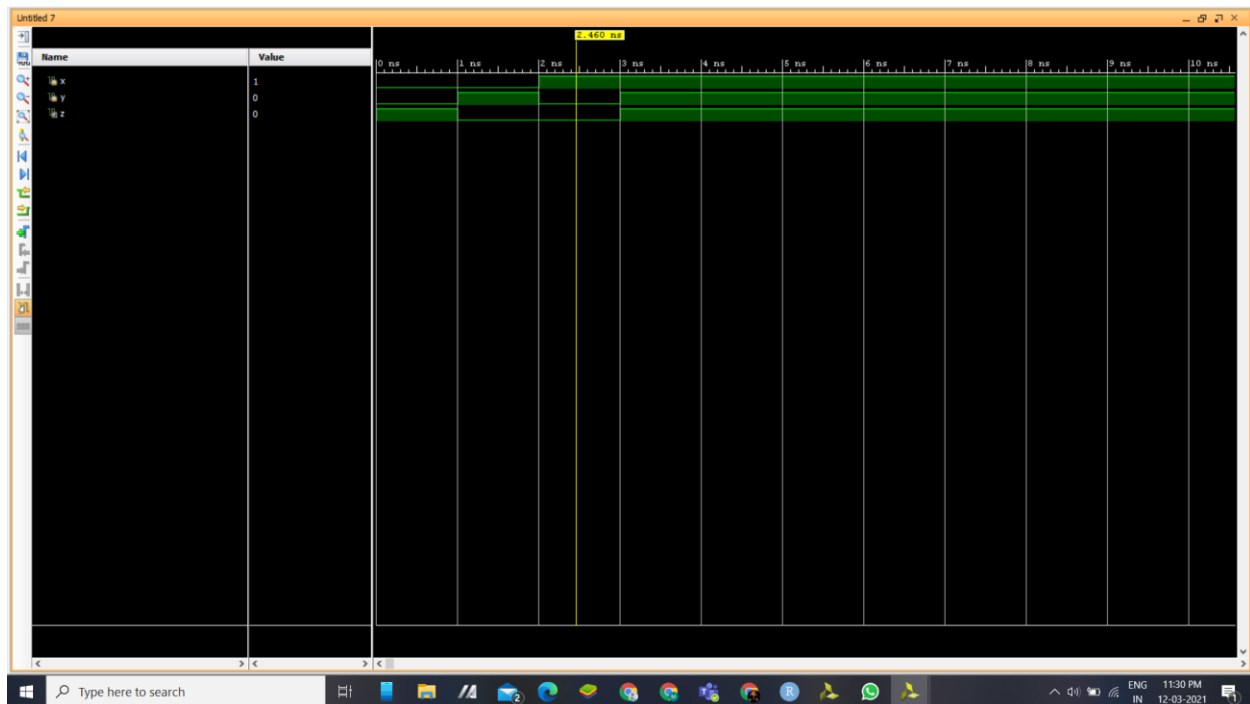
Xor gate:





Xnor gate:





Conclusion:

Now I got full knowledge about the verilog codes of the logic gates and there are three modelling styles also in that in this lab only two I learned(dataflow and structural)

1)dataflow modelling

2)behavioral modelling

3)structural modelling

There are three modelling styles in that two was very well known and I can do any codes for logic gates

Finally in this lab I came to know about

- Truth tables
- Logic gate diagrams
- Boolean expressions for logic gates
- Different types of modelling
- Output analysis

Reference links:

[circuitverse](https://www.circuitverse.com/)