

DIGITAL LOGIC DESIGN LAB

REPORT

ECE 1003 L 21-22

WIN 2022-23

DECLARATION BY THE CANDIDATE

I, SWARNLATA, solemnly declare that the project report is based on my own work carried out during the course of our study under the supervision of Chafle pratiksha vasantrao. I assert the statements made and conclusions drawn are an outcome of my research work.

I further certify that

- I. The work contained in the report is original and has been done by me under the general supervision of my supervisor.
- II. The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or any other University of India or abroad.
- III. I have followed the guidelines provided by the university in writing the report.

Name: SWARNLATA

Roll No.: 22BCE8591

Experiment List

Sr. No	Experiment Name
1	DESIGN AND VERIFICATION OF VARIOUS LOGIC GATES
2	
3	
4	
5	
6	
7	
8	
9	

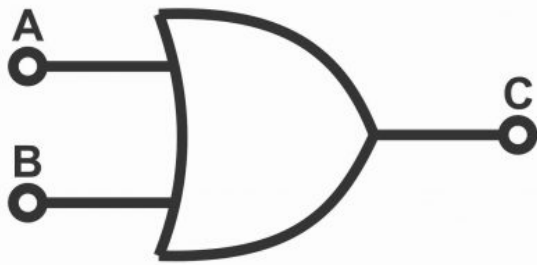
Experiment 1

OR GATE:

Objective: Verification of OR gate..

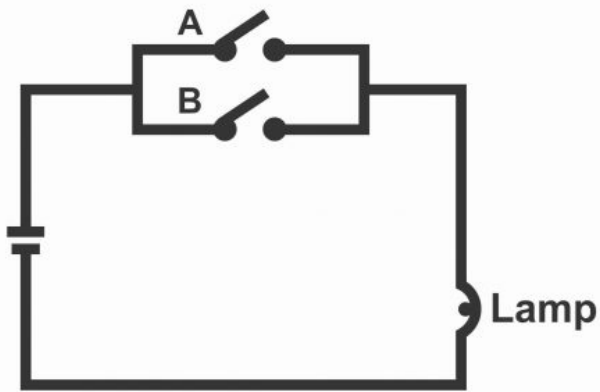
Theory:. TO evaluate or gate using eda playground $Y=(A.B)$ The OR Gate may have two or more inputs and only output. The output assumes the logic 1 state, even if one of its inputs is at logic 1 state. Its output assumes logic 0 state, only when each of its input logic 0 state.

Block Diagram: -



$$A+B=C$$

(a)



(b)

Fig 2.4

Implementation and output:

The screenshot shows the EDA Playground interface. On the left, there are tabs for "Languages & Libraries", "Tools & Simulators", and "Community". The main area is divided into two panels: "testbench.sv" and "design.sv".

testbench.sv:

```
1 // 228CE20237
2 module tb_orgate_puneeth;
3   reg x,y;
4   wire z;
5
6   or_gate_logic_puneeth_feb a1(x,y,z);
7
8
9   initial begin
10    $dumpfile("dump.vcd");
11    $dumpvars;
12    #100 $finish;
13  end
14
15  initial
16  begin
17    x = 0; y = 0;
18    $monitor("x=%0b y=%0b z=%0b", x, y, z); //display truth table
19    #5 x=0;y=1;
20    #5 x=1;y=0;
21    #5 x=1;y=1;
22  end
23 endmodule
```

design.sv:

```
1 module or_gate_logic_puneeth_feb(x,y,z);
2   input x,y;
3   output z;
4   assign z=(x | y);
5 endmodule
```

The bottom panel shows the simulation log:

```
[2023-02-20 03:50:15 EST] iverilog '-wall' design.sv testbench.sv && unbuffer vvp a.out
VCD info: dumpfile dump.vcd opened for output.
x=0 y=0 z=0
x=0 y=1 z=1
x=1 y=0 z=1
x=1 y=1 z=1
Finding VCD file...
./dump.vcd
[2023-02-20 03:50:15 EST] Opening EPWave...
```

The EPWave window is visible in the bottom right corner.

Waveform: -

OBJECTIVE : verification of AND gate

THEORY: TO evaluate AND gate ($y=A.B$) using eda playground. The AND gate is a basic digital logic gate that implements logical conjunction (\wedge) from mathematical logic – AND gate behaves according to the truth table. A HIGH output (1) results only if all the inputs to the AND gate are HIGH (1). If not all inputs to the AND gate are HIGH, LOW output results.

BLOCK DIAGRAM:

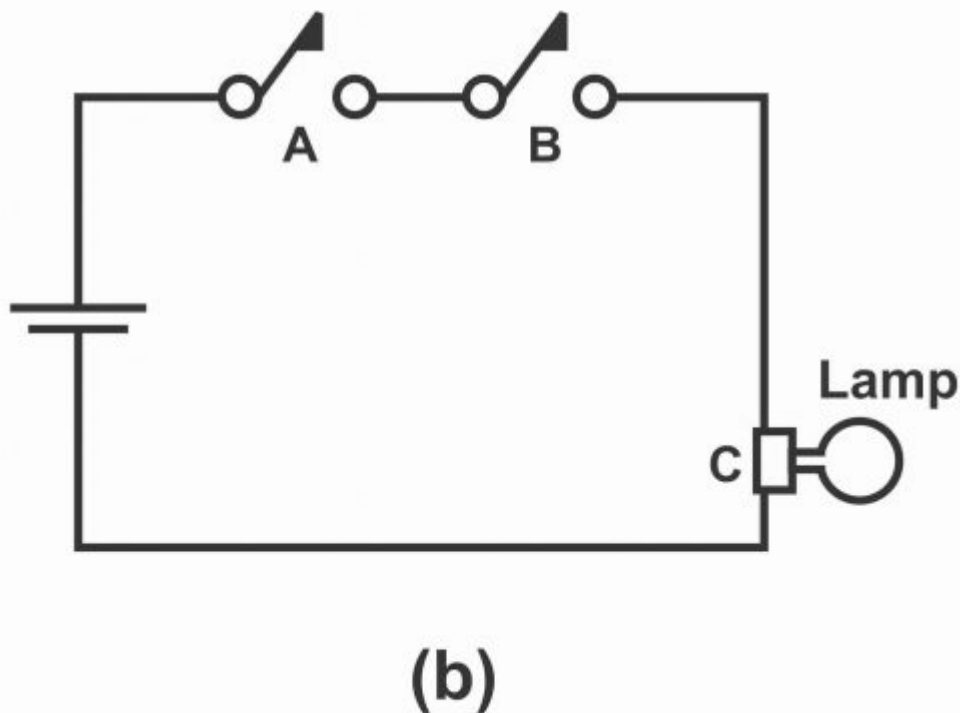
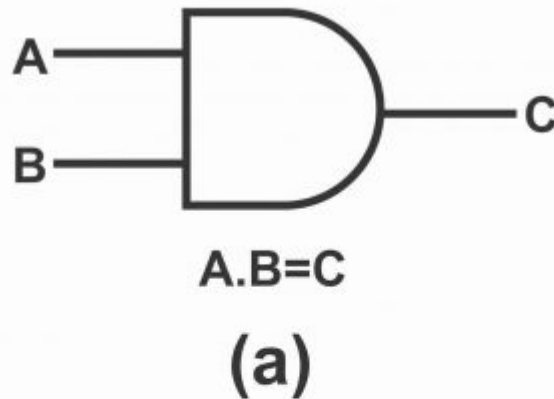


Fig 2.11

IMPLEMENTATION AND OUTPUT:

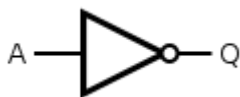
A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

NOT GATE

OBJECTIVE : VERIFICATION OF NOT GATE

THEORY: to verify not gate using eda playground. The NOT gate is an electronic circuit that produces an inverted version of the input at its output. It is also known as an inverter. If the input variable is A, the inverted output is known as NOT A. This is also shown as A', or A with a bar over the top, as shown at the outputs.

BLOCK DIAGRAM:



IMPLEMENTATION AND OUTPUT:

EDA playground

Run Save*

Brought to you by DOULOS

Languages & Libraries

Testbench + Design

SystemVerilog/Verilog

UVM / OVM

None

Other Libraries

None

OVL 2.8.1

SVUnit 2.11

Enable TL-Verilog

Enable Easier UVM

Enable VUnit

Tools & Simulators

Icarus Verilog 0.9.7

Compile Options

-Wall

Run Options

Run Options

Open EPWave after run

Download files after run

Examples

Community

Collaborate

Forum

Follow @edaplayground

testbench.sv

```
1 //22BCE20237
2 module NOT_Gate_tb;
3   reg A;
4   wire Y;
5
6   NOT_Gate inst(.A(A), .Y(Y));
7
8   initial begin
9     $dumpfile("dump.vcd");
10    $dumpvars;//dumping all variables
11    #1000 $finish;
12  end
13
14  initial begin
15    A=1'b0;
16    #50 $finish;
17  end
18  always #5 A=~A;
19  always @(Y)
20    $display("time=%0t :A=%b\t output:Y=%b", $time,A,Y);
21
22 endmodule
23
```

design.sv

```
1 //22BCE20237
2
3 module NOT_Gate(
4   input A,
5   output Y);
6
7   assign Y = ~A;//Y is opposite of A
8
9 endmodule
```

Log

time= 15t :A=1 output:Y=0

time= 20t :A=0 output:Y=1

time= 25t :A=1 output:Y=0

time= 30t :A=0 output:Y=1

time= 35t :A=1 output:Y=0

time= 40t :A=0 output:Y=1

time= 45t :A=1 output:Y=0

time= 50t :A=0 output:Y=1

Finding VCD file...

./dump.vcd

[2023-02-20 04:41:51 EST] Opening EPWave...

Done

EPWave

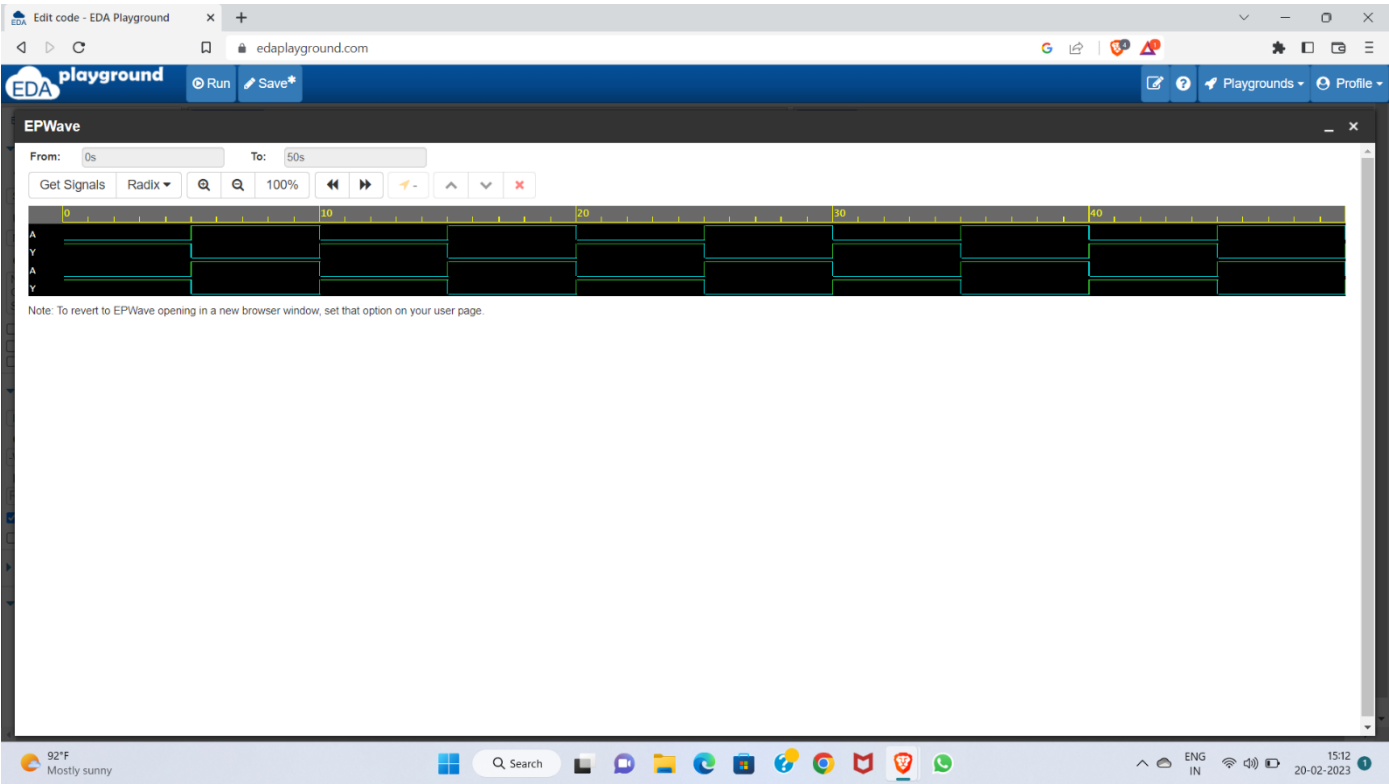
92°F Mostly sunny

Search

ENG IN

15:12 20-02-2023

WAVE FORM:



OBSERVATION:

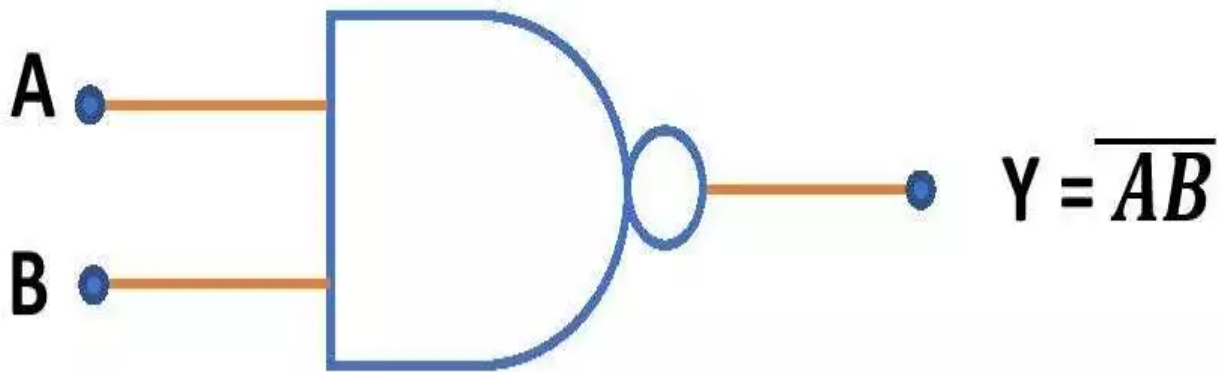
Input	Output
A	Y
0	1
1	0

NAND GATE

OBJECTIVE: VERIFICATION OF NAND GATE

THEORY: TO evaluate nand gate using eda play ground. In digital electronics, a NAND gate is a logic gate which produces an output which is false only if all its inputs are true; thus its output is complement to that of an AND gate. A LOW output results only if all the inputs to the gate are HIGH; if any input is LOW, a HIGH output results.

BLOCK DIAGRAM:



IMPLEMENTATION AND OUTPUT:

The screenshot shows the EDA Playground interface. On the left, there are tabs for 'Languages & Libraries', 'Tools & Simulators', and 'Community'. The main area is divided into two code editors: 'testbench.sv' and 'design.sv'. The 'testbench.sv' editor contains the following code:

```
//22BCE20237
module tb_nandgate_puneeth;
  reg x,y;
  wire z;

  nand_gate_logic_puneeth_feb a1(x,y,z);

  initial begin
    $dumpfile("dump.vcd");
    $dumpvars;//dumping all variables
    #100 $finish;
  end

  initial
  begin
    x = 0;y=0;
    $monitor ("%x=%0b y=%0b z=%0b",x,y,z);//display truthtable
    #5 x=0;y=1;
    #5 x=1;y=0;
    #5 x=1;y=1;
  end
endmodule
```

The 'design.sv' editor contains the following code:

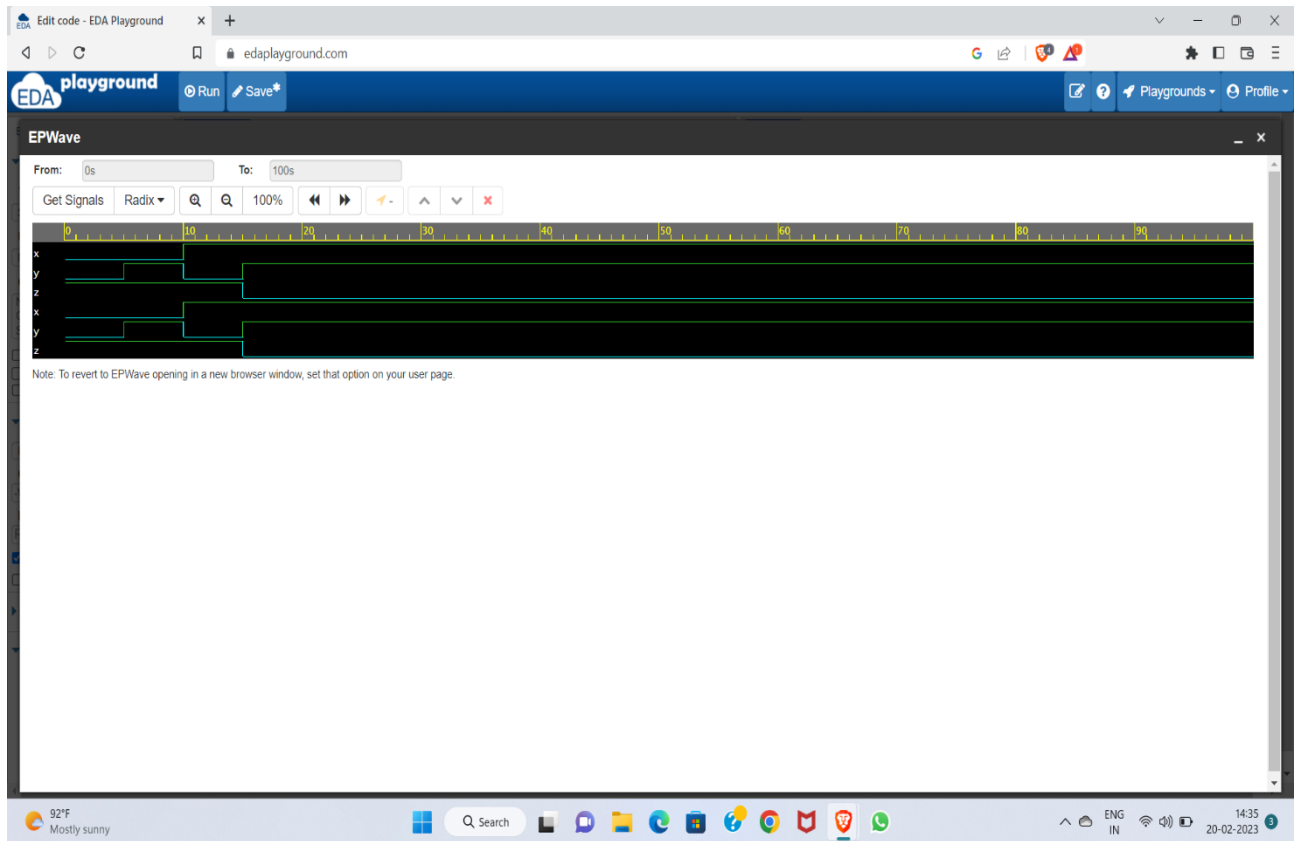
```
// Code your design here
module nand_gate_logic_puneeth_feb(x,y,z);
  input x,y;
  output z;
  assign z = ~(x&y);
endmodule
```

Below the code editors, there is a 'Log' tab showing the simulation output:

```
[2023-02-20 04:02:56 EST] iverilog '-wall' design.sv testbench.sv && unbuffer vvp a.out
VCD info: dumpfile dump.vcd opened for output.
x=0 y=0 z=1
x=0 y=1 z=1
x=1 y=0 z=1
x=1 y=1 z=0
Finding VCD file...
./dump.vcd
[2023-02-20 04:02:56 EST] Opening EPWave...
Done
```

At the bottom right, there is a tab for 'EPWave' which is currently closed.

WAVE FORM:



OBSERVATIONS:

Input		Output
A	B	$Y = \overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

V

NOR GATE

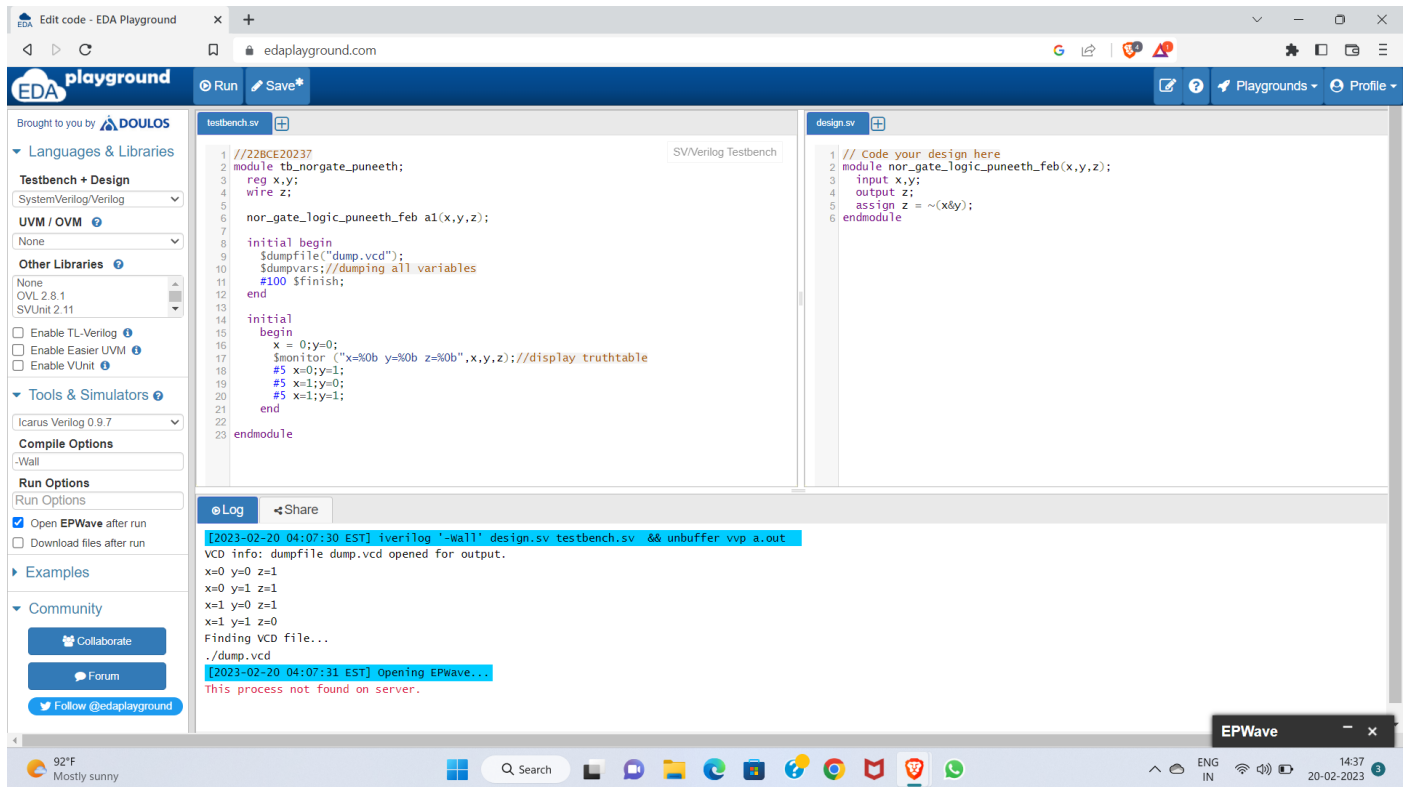
OBJECTIVE : verification of nor gate

THEORY: TO verify nor gate using eda playground. The NOR gate is a digital logic gate that implements logical NOR - it behaves according to the truth table to the right. A HIGH output results if both the inputs to the gate are LOW; if one or both input is HIGH, a LOW output results. NOR is the result of the negation of the OR operator.

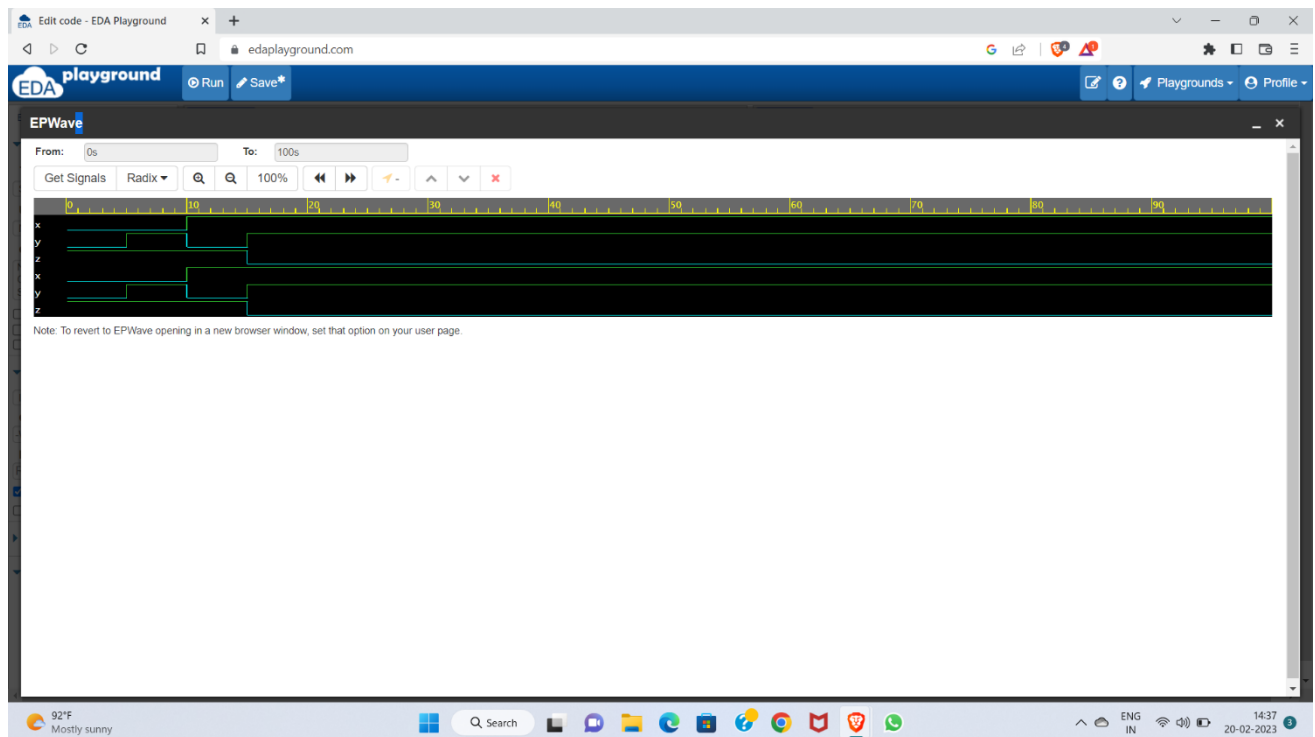
BLOCK DIAGRAM:



IMPLEMENTATION AND OUTPUT:



WAVE FORM:



OBSERVATIONS:

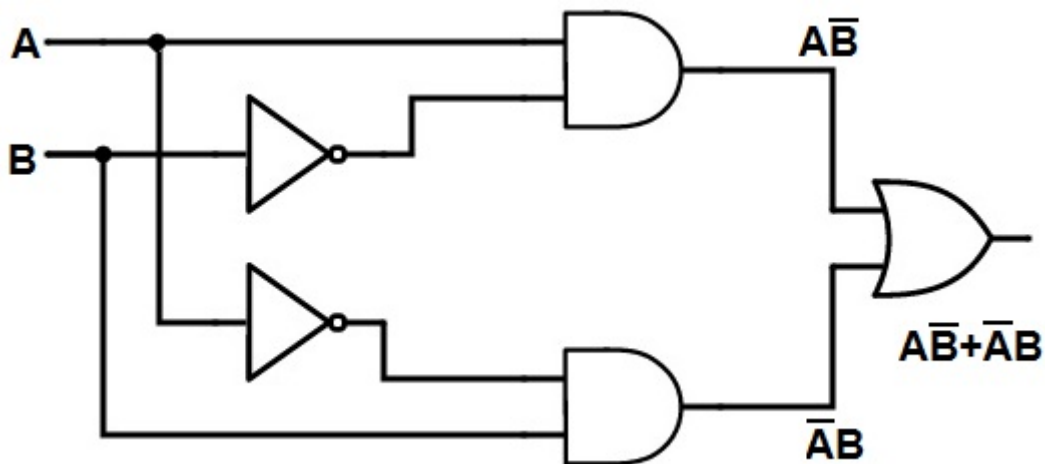
Input		Output
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

XOR GATE

OBJECTIVE: verification of xor gate

THEORY: evaluation of xor gate using eda playground. The output is "true" if either, but not both, of the inputs are "true." The output is "false" if both inputs are "false" or if both inputs are "true."

BLOCK DIAGRAM:



IMPLEMENTATION AND OUTPUT:



EX-OR (X-OR) Gate Truth Table

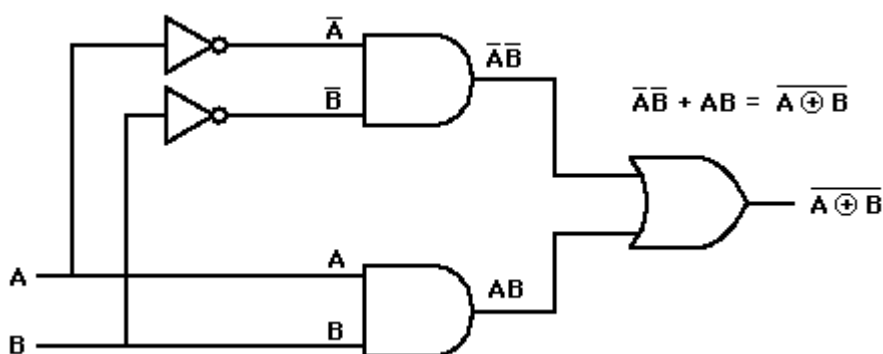
Inputs		Output $X = A \oplus B$
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

XNOR GATE

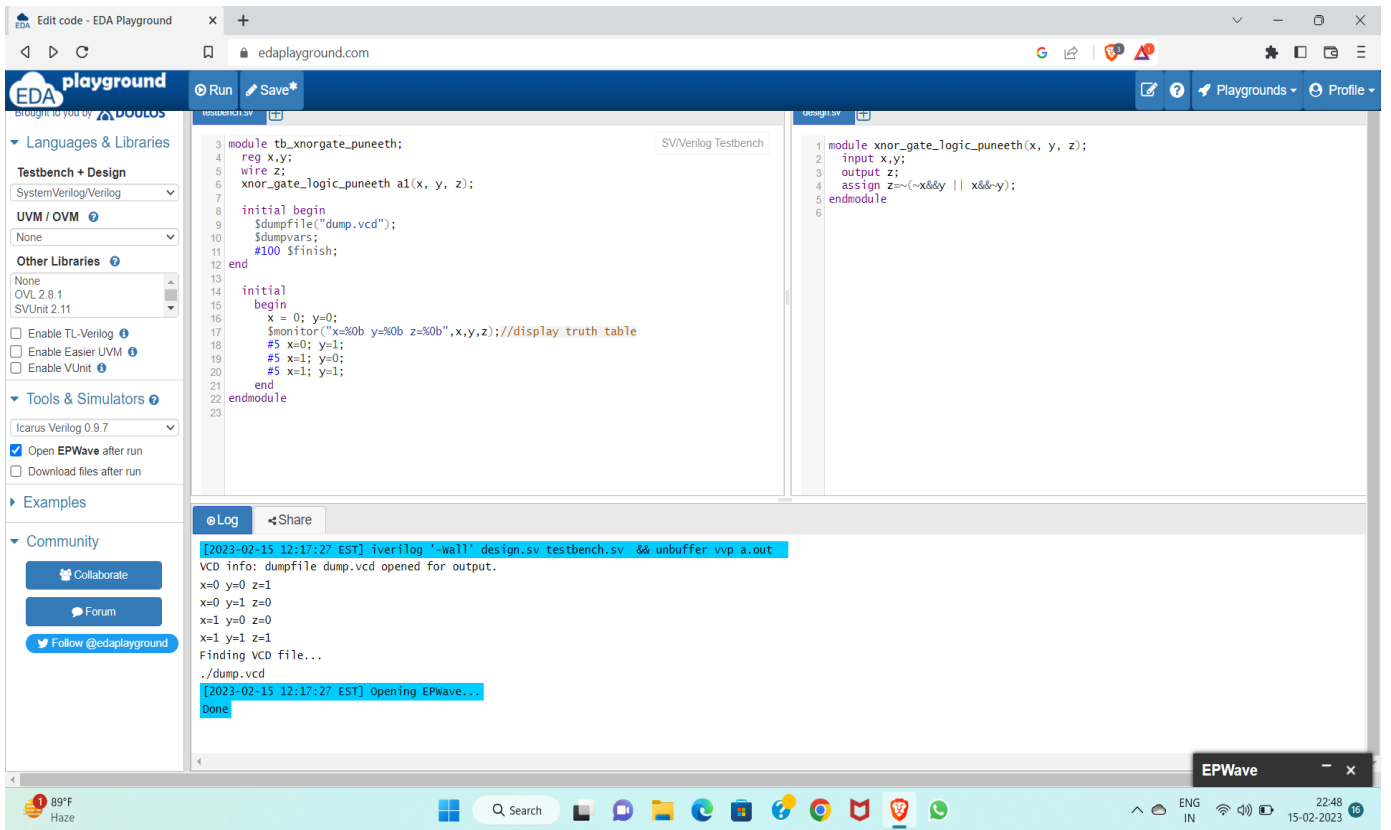
OBJECTIVE: Verification of xnor gate

THEORY: To evaluate xnor gate using eda playground. The XNOR (exclusive-NOR) is a combination XOR gate followed by an inverter. Its output is "true" if the inputs are the same, and "false" if the inputs are different.

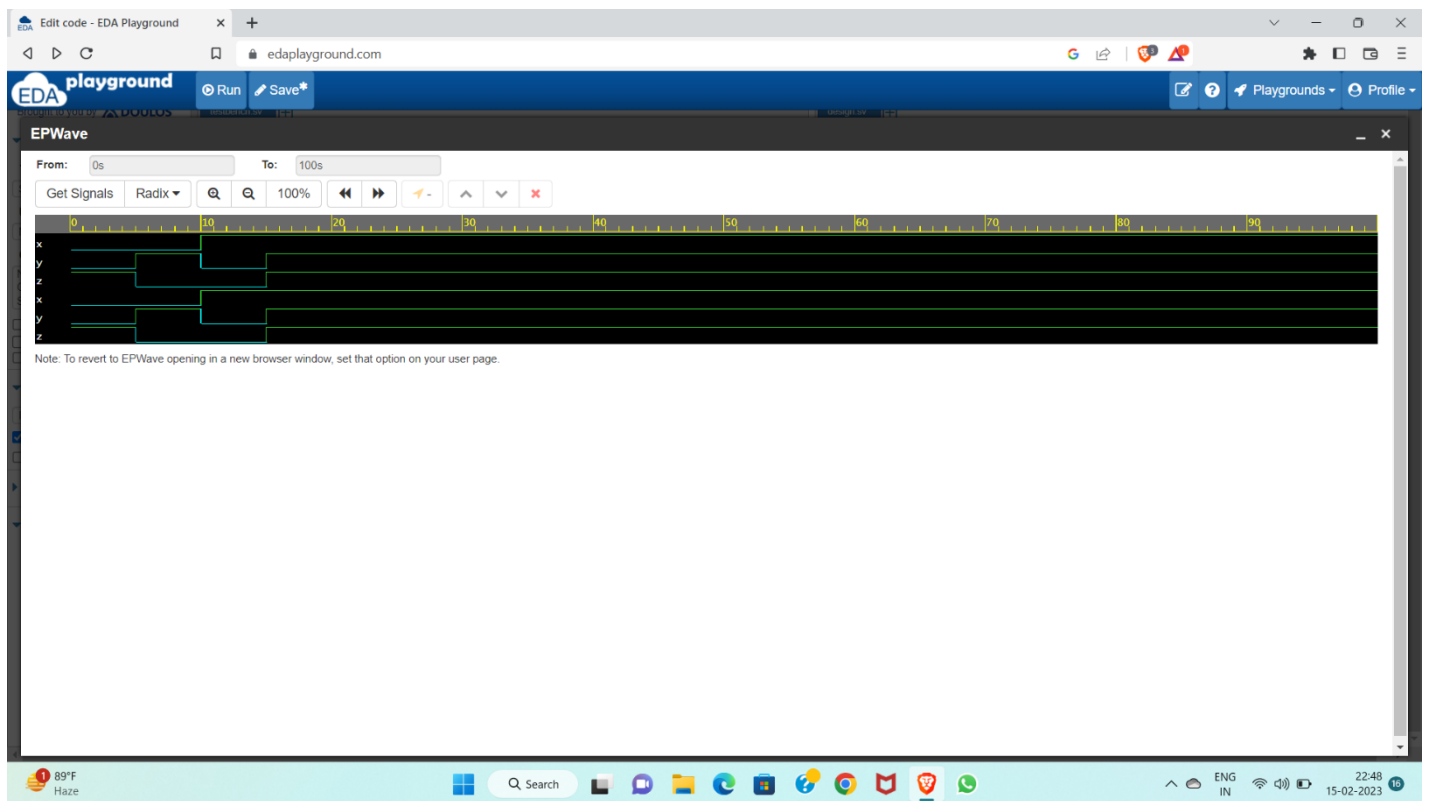
BLOCK DIAGRAM:



IMPLEMENTATION AND OUTPUT:



WAVE FORM:



OBSERVATIONS:

EX-NOR (X-NOR) Gate Truth Table

Inputs		Output $X = \overline{A \oplus B}$
A	B	
0	0	1
0	1	0
1	0	0
1	1	1