EXP NO: 01

VERIFICATION OF LOGIC GATES

AIM:

To develop the source code for logic gates by using VERILOG and obtain the simulation.

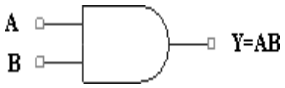
ALGORITM:

Step1: Define the specifications and initialize the design.
Step2: Write the source code in VERILOG.
Step3: Check the syntax and debug the errors if found, obtain the synthesis report.
Step4: Verify the output by simulating the source code.
Step5: Write all possible combinations of input using the test bench.
Step6: Obtain the place and route report.

LOGIC DIAGRAM:

AND GATE:

LOGIC DIAGRAM:

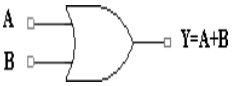


TRUTH TABLE:

A	B	Y=AB
0	0	0
0	1	0
1	0	0
1	1	1

OR GATE:

LOGICDIAGRAM

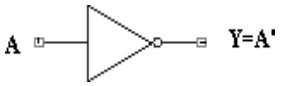


TRUTH TABLE:

A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1

NOT GATE:

LOGIC DIAGRAM:

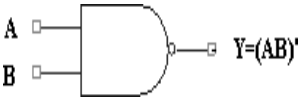


TRUTH TABLE:

A	Y=A’
0	1
1	0

NAND GATE:

LOGICDIAGRAM

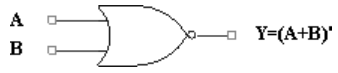


TRUTH TABLE

A	B	Y=(AB)’
0	0	1
0	1	1
1	0	1
1	1	0

NOR GATE:

LOGIC DIAGRAM:

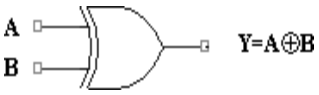


TRUTH TABLE:

A	B	Y=(A+B)’
0	0	1
0	1	0
1	0	0
1	1	0

XOR GATE:

LOGICDIAGRAM

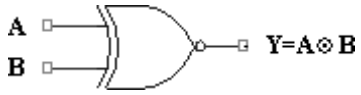


TRUTH TABLE

A	B	Y=A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

XNOR GATE:

LOGIC DIAGRAM:



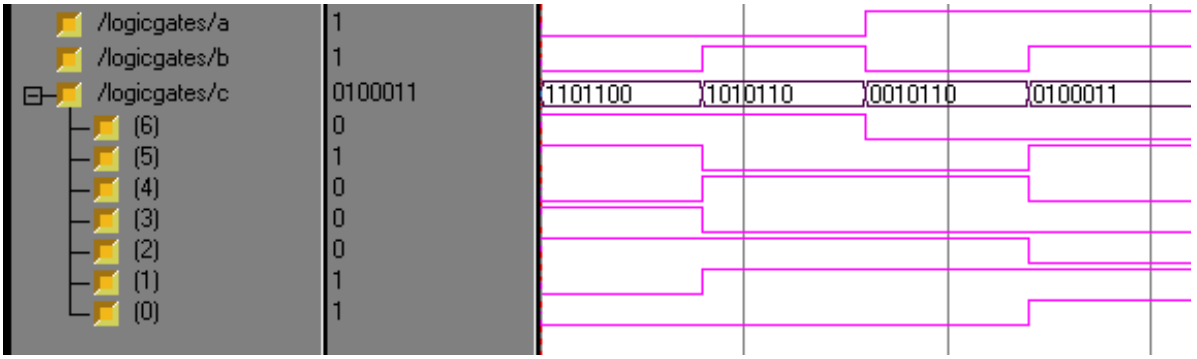
TRUTH TABLE:

A	B	Y=A⊙B
0	0	1
0	1	0
1	0	0
1	1	1

VERILOG SOURCE CODE:

```
module logicgates1(a, b, c);
   input a;
   input b;
   OUTPUT: [6:0] c;
         assign c[0]= a & b;
         assign c[1]= a | b;
         assign c[2]= ~(a & b);
         assign c[3]= ~(a | b);
         assign c[4]= a ^ b;
         assign c[5]= ~(a ^ b);
         assign c[6]= ~ a;
end module
```

Simulation output:



==========

RESULT:

Thus, the outputs of Basic Logic Gates are verified by simulating and synthesizing the VERILOG code.