# Digital System Design

## Lab01

**Spring 2025**


Submitted by: **Mohsin Sajjad**

Registration No: **22pwsce2149**

Class Section: **A**


"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Student Signature: _____

Submitted to:

**Engr. Faheem Jan**

Month Day, Year (16 02, 2025)


Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

# LAB No 1
## INTRODUCTION TO MODELSIM AND GATE LEVEL MODELING

**Objectives:**

Introduction to MODELSIM

**Software used:**

MODELSIM

**MODELSIM:**

MODELSIM is a simulator which can be used for the simulations of both VHDL and Verilog HDL. It has the following interface.

**TASKS:**

1. Implement a buffer at the gate level.

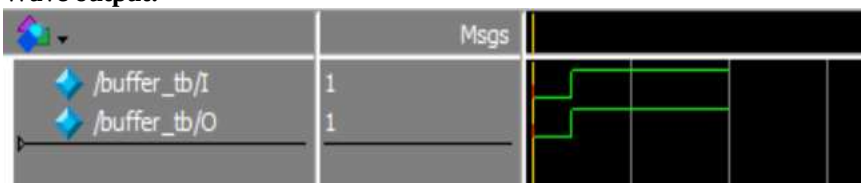
**CODE:**

```
module buffer(I,O);
input I;
output O;
buf b(O,I);
endmodule

module buffer_tb();
reg I;
wire O;
buffer inst(I,O);
initial begin  //for multiple lines
I=0;
#20 $display("I = %b",I,"O = %b",O);

I=1;
#20 $display("I = %b",I,"O = %b",O);
end
endmodule
```

**Truth table:**

```
VSIM 4> run
# I = 00 = 0
# I = 10 = 1
```

**Wave output:**



2. Implement an inverter at the gate level.

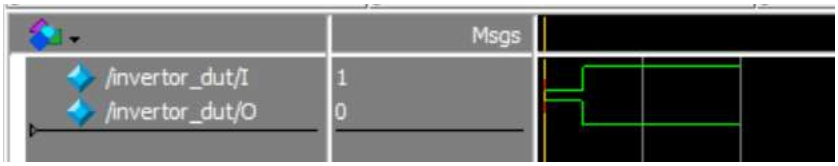
Code:

```
Ln#
 1  module invertor(I,O);
 2  input I;
 3  output O;
 4  not a(O,I);
 5  endmodule
 6
 7  module invertor_dut();
 8  reg I;
 9  wire O;
10  invertor b(I,O);
11  initial
12  begin
13  I=0;
14  #20 $display("I=%b",I,"O=%b",O);
15  I=1;
16  #20 $display("I=%b",I,"O=%b",O);
17  end
18  endmodule
```

Truth Table:

```
VSIM 14> run
# I=0O=1
# I=1O=0
```

Wave output:



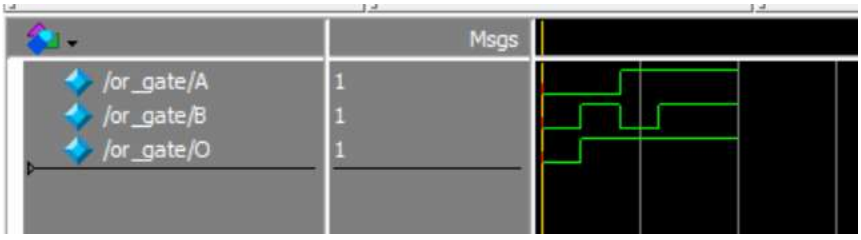

3. Implement an OR gate using a NAND gates.

Code:

```
module orgate(A, B, O);
input A, B;
output O;
wire nandout1, nandout2;
nand n1(nandout1, A);
nand n2(nandout2, B);
nand n3(O, nandout1, nandout2);

endmodule

module or_gate();
reg A, B;
wire O;
orgate x(A, B, O);

initial begin
    A = 0; B = 0;
    #20 $display("A = %b, B = %b, O = %b", A, B, O);

    A = 0; B = 1;
    #20 $display("A = %b, B = %b, O = %b", A, B, O);

    A = 1; B = 0;
    #20 $display("A = %b, B = %b, O = %b", A, B, O);

    A = 1; B = 1;
    #20 $display("A = %b, B = %b, O = %b", A, B, O);
end

endmodule
```

```
VSIM 23> run
# A = 0, B = 0, O = 0
# A = 0, B = 1, O = 1
# A = 1, B = 0, O = 1
# A = 1, B = 1, O = 1
```

Wave output:



4. Implement the following equation where z is output and x1, x2, x3, x4, and x5 are inputs of the circuit.

z = ( y1 + y2 )'

y1 = x1.x2

y2 = (x3.x4.x5)

```
module equ(Z,x1,x2,x3,x4,x5);
input x1,x2,x3,x4,x5;
output Z;
wire y1;
wire y2;
and n1(y1,x1,x2);
and n2(y2,x3,x4,x5);
nor n3(Z,y1,y2);
endmodule

module equ_tb();
reg x1,x2,x3,x4,x5;
wire Z;
equ ins(Z,x1,x2,x3,x4,x5);
initial begin
x1=0;
x2=0;
x3=0;
x4=0;
x5=0;
#20 $display("x1 = %b",x1, "x2 = %b",x2,"x3 = %b",x3,"x4 = %b",x4,"x5 = %b",x5,"Z = %b",Z);

x1=0;
x2=0;
x3=0;
x4=0;
x5=1;
#20 $display("x1 = %b",x1, "x2 = %b",x2,"x3 = %b",x3,"x4 = %b",x4,"x5 = %b",x5,"Z = %b",Z);

x1=0;
x2=0;
x3=0;
x4=1;
x5=0;
#20 $display("x1 = %b",x1, "x2 = %b",x2,"x3 = %b",x3,"x4 = %b",x4,"x5 = %b",x5,"Z = %b",Z);
```

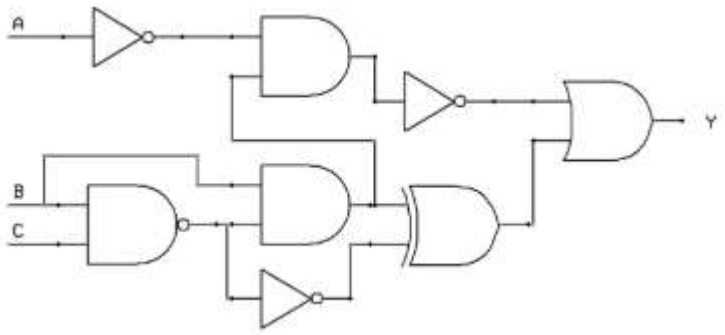
Truth Table:

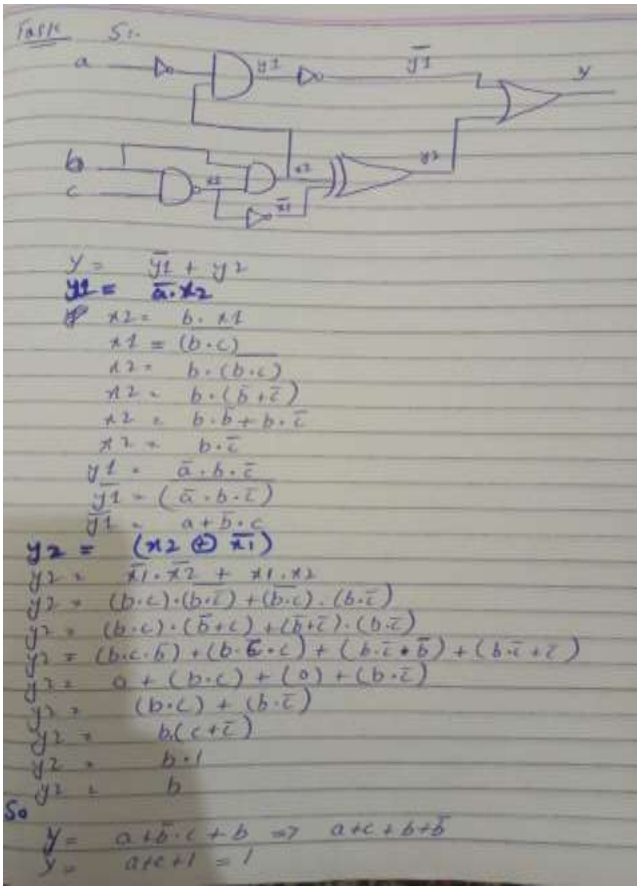
```
add wave -position insertpoint sim:/equ_tb/*
VSIM 19> run
# x1 = 0x2 = 0x3 = 0x4 = 0x5 = 0Z = 1
# x1 = 0x2 = 0x3 = 0x4 = 0x5 = 1Z = 1
# x1 = 0x2 = 0x3 = 0x4 = 1x5 = 0Z = 1
# x1 = 0x2 = 0x3 = 0x4 = 1x5 = 1Z = 1
```

Wave output:


Msgs
/equ_tb/x1
/equ_tb/x2
/equ_tb/x3
/equ_tb/x4
/equ_tb/x5
/equ_tb/Z
0
0
0
1
1
1

5.


A
B
C
Y


a
b
c
y
y = ȳ1 + y2
ȳ1 = ā · x2
x2 = b · x1
x1 = (b · c)
x2 = b · (b · c)
x2 = b · (b̄ + c̄)
x2 = b · b̄ + b · c̄
x2 = b · c̄
y1 = ā · b · c̄
ȳ1 = (ā · b · c̄)
ȳ1 = a + b̄ · c
y2 = (x2 ⊕ x1)
y2 = x̄1 · x̄2 + x1 · x2
y2 = (b · c) · (b · c̄) + (b · c) · (b · c̄)
y2 = (b · c) · (b̄ + c) + (b̄ + c̄) · (b · c̄)
y2 = (b · c · b̄) + (b · b̄ · c) + (b · c̄ · b̄) + (b · c̄ · c)
y2 = 0 + (b · c) + (0) + (b · c̄)
y2 = (b · c) + (b · c̄)
y2 = b(c + c̄)
y2 = b · 1
y2 = b
So
y = a + b̄ · c + b ⇒ a + c + b + b̄
y = a + c + 1 = 1

```
module logic_circuit(output Y, input A, B, C);
    wire A_not, X1, X2, X2_not, Y1, Y2, Y1_not;

    // Inverters
    not n1(A_not, A);
    not n2(X1_not, X1);

    // AND Gates
    nand a2(X1,B, C);
    and a3(X2,X1,B);
    and a1(Y1, A_not, X2);
    // XOR Gate
    xor x1(Y2, X2, X1_not);

    // Inverter for Y1
    not n3(Y1_not, Y1);

    // OR Gate
    or o1(Y, Y1_not, Y2);
endmodule

module logic_circuit_tb();
    reg A, B, C;
    wire Y;
    logic_circuit uut(Y, A, B, C);

    initial begin
        // Display header
        $display("A  B  C  |  Y");
        $display("----------------");

        // Test all possible input combinations
        A = 0; B = 0; C = 0; #10; $display("%b  %b  %b  |  %b", A, B, C, Y);
        A = 0; B = 0; C = 1; #10; $display("%b  %b  %b  |  %b", A, B, C, Y);
        A = 0; B = 1; C = 0; #10; $display("%b  %b  %b  |  %b", A, B, C, Y);
```

```
        A = 0; B = 1; C = 1; #10; $display("%b  %b  %b  |  %b", A, B, C, Y);
        A = 1; B = 0; C = 0; #10; $display("%b  %b  %b  |  %b", A, B, C, Y);
        A = 1; B = 0; C = 1; #10; $display("%b  %b  %b  |  %b", A, B, C, Y);
        A = 1; B = 1; C = 0; #10; $display("%b  %b  %b  |  %b", A, B, C, Y);
        A = 1; B = 1; C = 1; #10; $display("%b  %b  %b  |  %b", A, B, C, Y);

        $stop;
    end
endmodule
```

**Truth table:**

```
VSIM 27> run
# A  B  C  |  Y
# ----------------
# 0  0  0  |  1
# 0  0  1  |  1
# 0  1  0  |  1
# 0  1  1  |  1
# 1  0  0  |  1
# 1  0  1  |  1
# 1  1  0  |  1
# 1  1  1  |  1
# ** Note: $stop    : D:/Semes
#    Time: 80 ns  Iteration: 0
```

Wave output:


Msgs
/logic_circuit_tb/A
/logic_circuit_tb/B
/logic_circuit_tb/C
/logic_circuit_tb/Y
1
1
1
1