National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science
Department of Electrical Engineering

Faculty Member: Sir Arshad Nazir **Dated: 21st September 2023**

Semester: Fall 2023 **Section: BEE-14-D**

Group No.: 22

EE-221: Digital Logic Design

Lab 2: Introduction to Verilog

		PLO4/CLO4	PLO4/CLO4	PLO5/CLO5	PLO8/CLO6	PLO9/CLO7	
Name	**Reg. No**	**Viva / Lab Performance** **5 Marks**	**Analysis of data in Lab Report** **5 Marks**	**Modern Tool Usage** **5 Marks**	**Ethics and Safety** **5 Marks**	**Individual and Team Work** **5 Marks**	**Total marks Obtained** **25 Marks**
Arooj Fatima	**423365**						
Irfa Farooq	**412564**						
Ahmad Nasir	**409959**						
Haseeb Umar	**427442**						

Lab2: Introduction to Verilog, Gate-level/Behavioral Modeling and Hardware Implementation of Basic Logic Circuit

This Lab has been divided into two parts.

In first part you will be introduced to Verilog and Gate-Level Modeling.
The next part is the hardware implementation of a Boolean function given to you.

Objectives:

- ✓ Understand HDL and compare it with normal programming languages.
- ✓ Simulate Basic Gates using Verilog with ModelSim
- ✓ Write stimulus using Verilog
- ✓ Derive algebraic expression for a Boolean function from the given schematics.
- ✓ Hardware Implementation of Logic Circuit

Lab Instructions

- ✓ This lab activity comprises three parts, namely Pre-lab, Lab tasks, and post-Lab Viva session.
- ✓ The lab report will be uploaded on LMS three days before the scheduled lab date. The students will get a hard copy of lab report, complete the Pre-lab task before coming to the lab and deposit it with teacher/lab engineer for necessary evaluation. Alternately each group to upload completed lab report on LMS for grading.
- ✓ The students failing to submit Pre-Lab will not be allowed to perform Lab work.
- ✓ The students will start lab task and demonstrate design steps separately for stepwise evaluation (course instructor/lab engineer will sign each step after ascertaining functional verification)
- ✓ Remember that a neat logic diagram with pins numbered coupled with nicely patched circuit will simplify trouble-shooting process.
- ✓ After the lab, students are expected to unwire the circuit and deposit back components before leaving.
- ✓ The students will complete lab tasks and submit complete report to Lab Engineer before leaving lab. The Verilog tutorial part is non-printable and for reference only.
- ✓ There are related questions at the end of this activity. Give complete answers.

Pre-Lab Task: (To be done before coming to the lab) (2 marks)

1. **Read the manual Getting Started with Verilog and answer the following questions.**
   a) HDL stands for

   HDL Stands for Hardware Description Language.

   It is a low-level programming language basically used to design circuits and describe hardware.

   b) Two standard versions of HDL are

   VHDL (VHSIC Hardware Description Language) and Verilog HDL.

   c) Give the different levels of abstraction in Verilog HDL

1. Gate Level Modelling:

   It uses basic gates e.g., AND and their interconnection to describe the whole system.

2. Dataflow Modelling:

   It defines flow of data between different components and doesn't need interconnection between gates.

3. Behavioral Modelling

   It defines the overall system behavior e.g., input is x and y is the output.

4. Switch Level Modelling:

   It involves the use of transistors.

The abstraction level increases as we move from Gate to Behavioral Modeling.

Lab Tasks: (8 marks)

Lab Task 1: (2 marks)

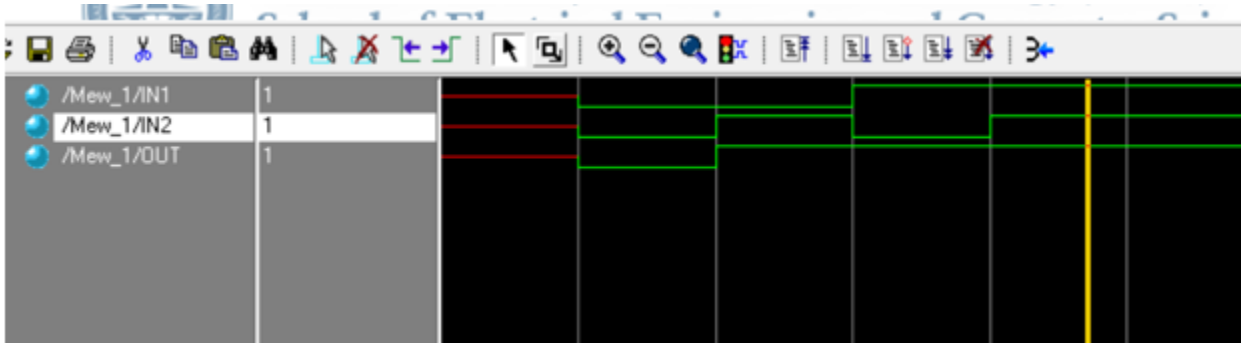
Model and simulate the basic gates i.e., NOT, AND & OR in Verilog (Gate level) using Modelsim. Compare the simulation waveform results with truth table in the space given below.

OR Gate Code:

```
module Lab_2(out,in1,in2);
input in1,in2;
output out;
or a1(out,in1,in2);
endmodule

module Mew1;
reg IN1,IN2;
wire OUT;
Lab_2 L1(OUT,IN1,IN2);
initial
begin
        #100 IN1=1'b0;IN2=1'b0;
        #100 IN1=1'b0;IN2=1'b1;
        #100 IN1=1'b1;IN2=1'b0;
        #100 IN1=1'b1;IN2=1'b1;
end
endmodule
```

Output Wave:



OR TRUTH TABLE:

Sr. no.	Input A	Input B	Output (A + B)
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	1

AND Gate Code:

```
module Lab_2(out,in1,in2);
input in1,in2;
output out;
and a1(out,in1,in2);
endmodule

module Mew_1;
reg IN1,IN2;
wire OUT;
Lab_2 L1(OUT,IN1,IN2);
initial
begin
        #100 IN1=1'b0;IN2=1'b0;
        #100 IN1=1'b0;IN2=1'b1;
        #100 IN1=1'b1;IN2=1'b0;
        #100 IN1=1'b1;IN2=1'b1;
end
endmodule
```

Output Waveform:

AND TRUTH TABLE:

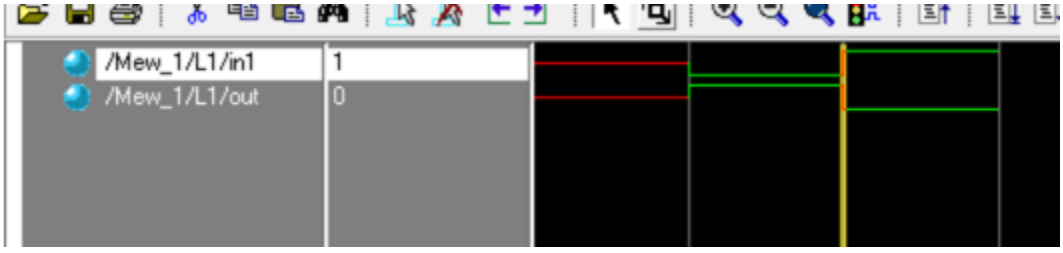
Sr. no.	Input A	Input B	Output (A.B)
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

NOT Gate Code:

```
module Lab_2(out,in1);
input in1;
output out;
not a1(out,in1);
endmodule

module Mew_1;
reg IN1;
wire OUT;
Lab_2 L1(OUT,IN1);
initial
begin
        #100 IN1=1'b0;
        #100 IN1=1'b1;
end
endmodule
```

Output Waveform:



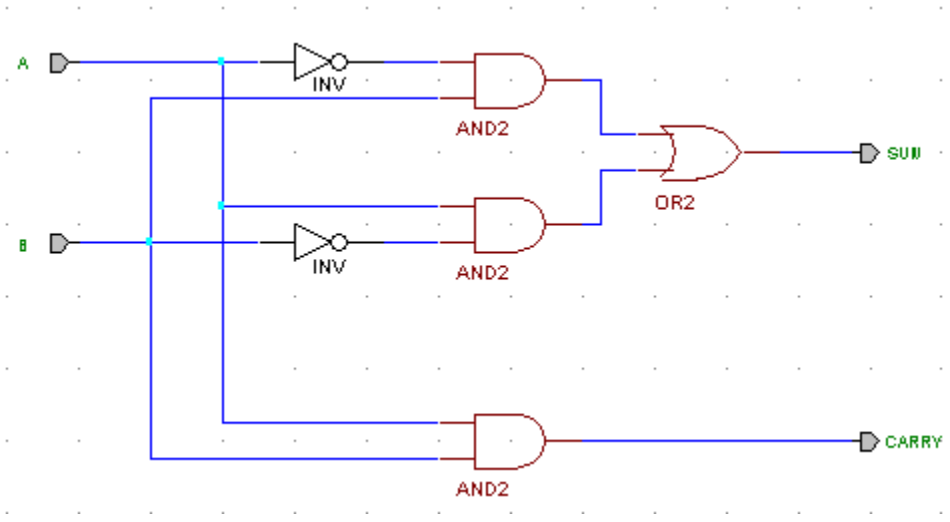
NOT TRUTH TABLE:

Sr. no.	Input A	Output A'
1	0	1
2	1	0

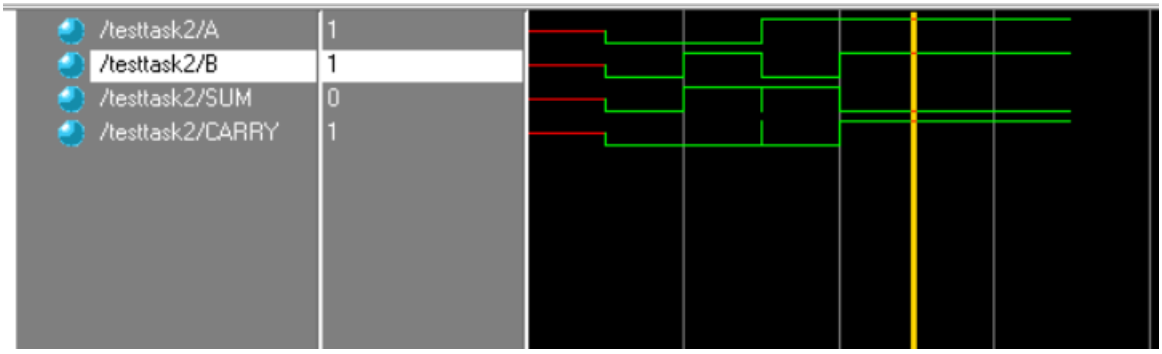
Lab Task 2: (2 marks)

Write the Verilog Code using Gate Level modeling for the following circuit. List the code for design as well as stimulus below:

Simulate the circuit below on Proteus and perform it on hardware.



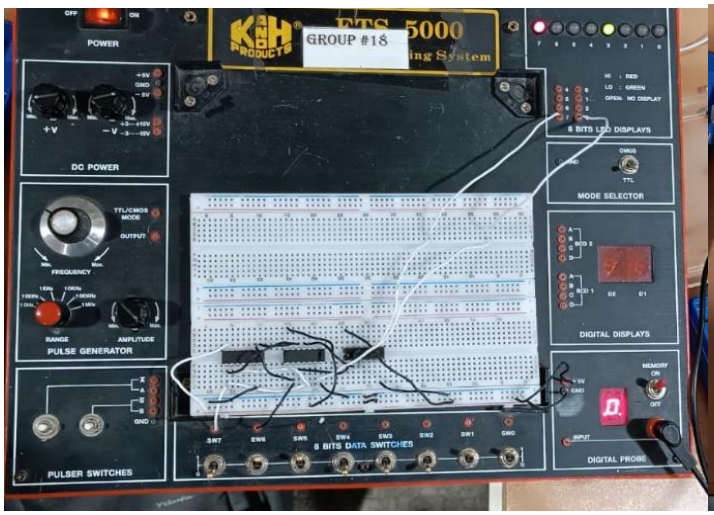
ModelSim Output Wave:

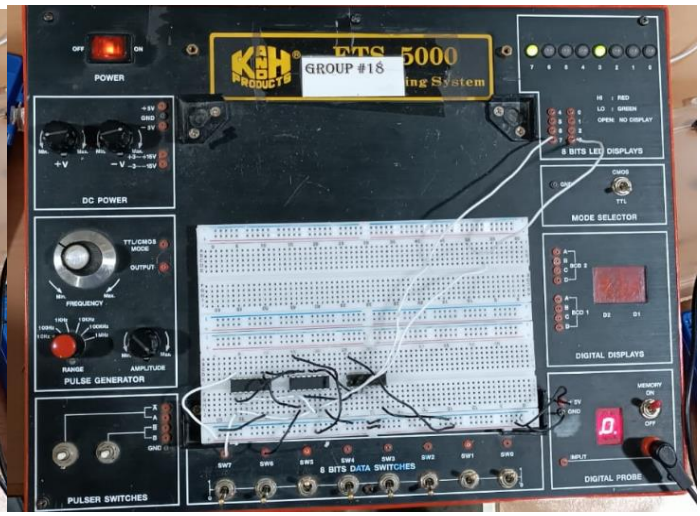


Design and Stimulus Code:

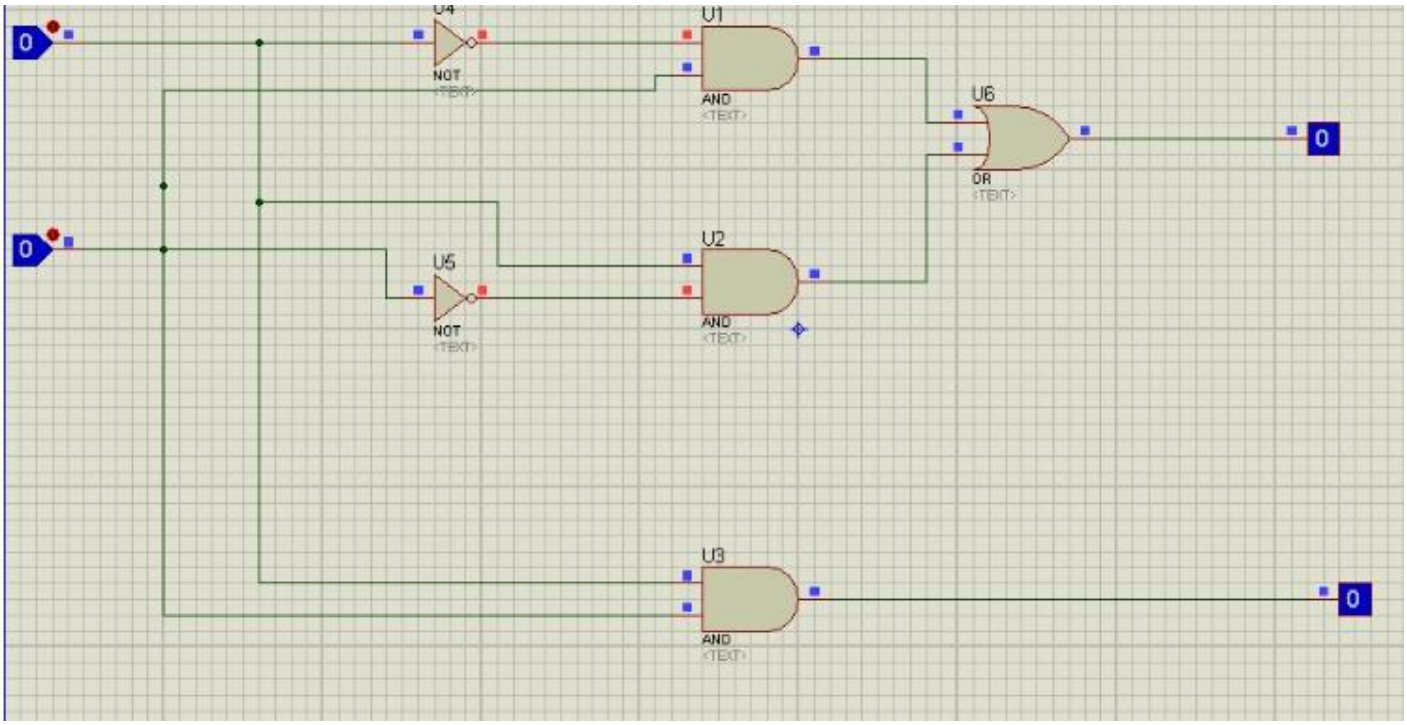
```
 1 module circuit(sum,carry,a,b);
 2 input a,b;
 3 output sum,carry;
 4 wire w1,w2,w3,w4;
 5
 6 not not1(w1,a);
 7 not not2(w2,b);
 8 and and1(w3,w1,b);
 9 and and2(w4,w2,a);
10 or or1(sum,w3,w4);
11 and and3(carry,a,b);
12 endmodule
13
14 module testtask2;
15 reg A,B;
16 wire SUM,CARRY;
17 circuit task2 (SUM,CARRY,A,B);
18 initial
19 begin
20              #100 A=1'b0;B=1'b0;
21              #100 A=1'b0;B=1'b1;
22              #100 A=1'b1;B=1'b0;
23              #100 A=1'b1;B=1'b1;
24  end
25  endmodule
```

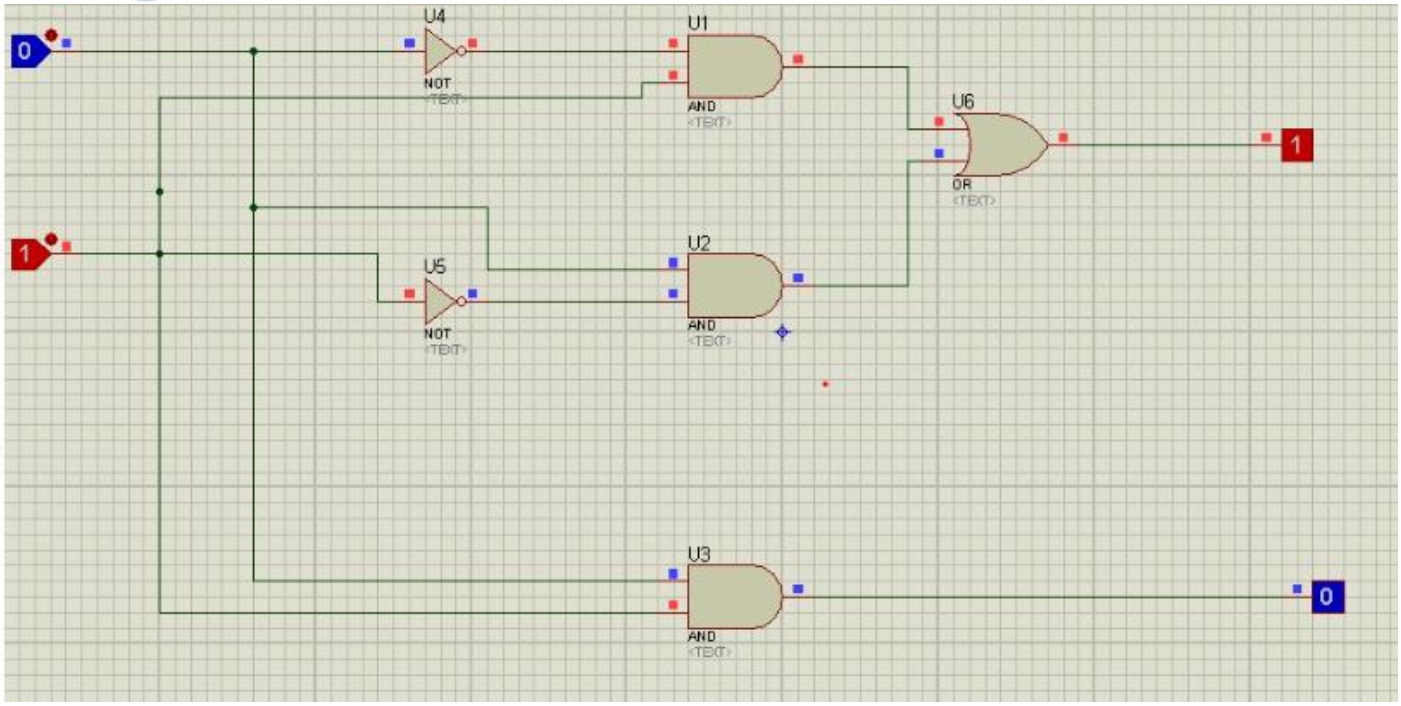
Hardware:

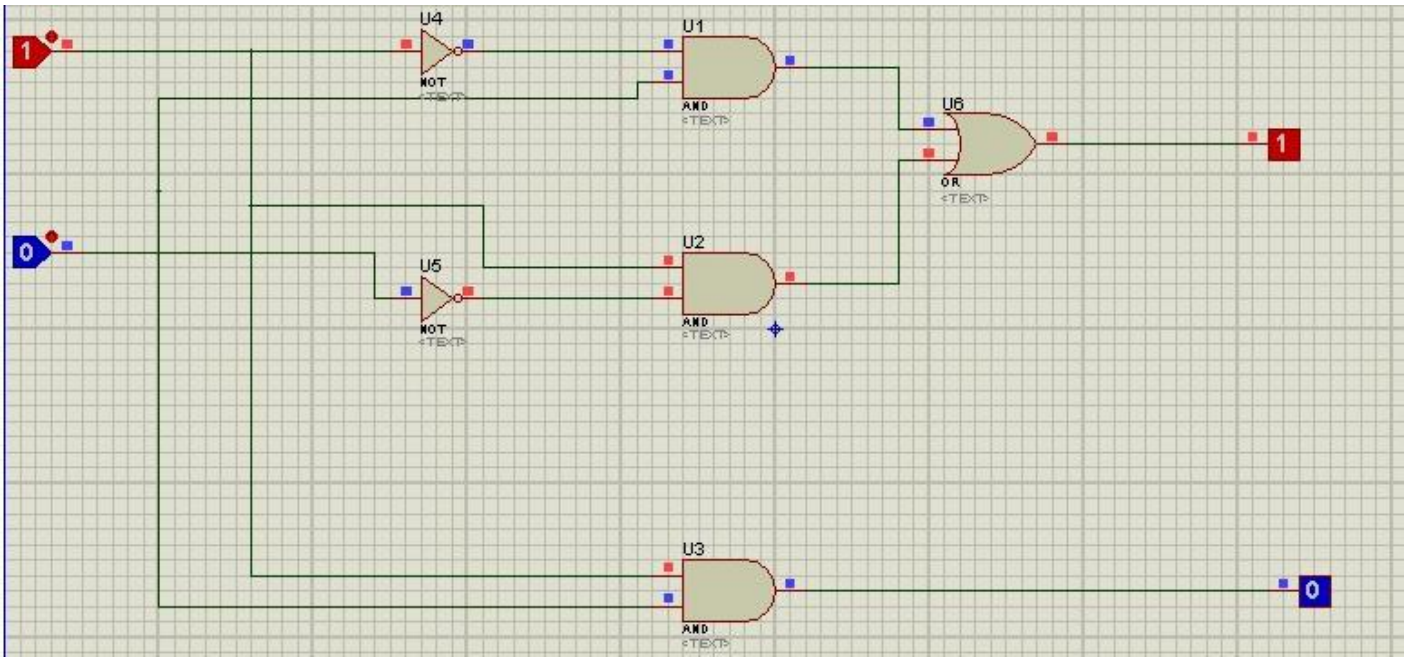


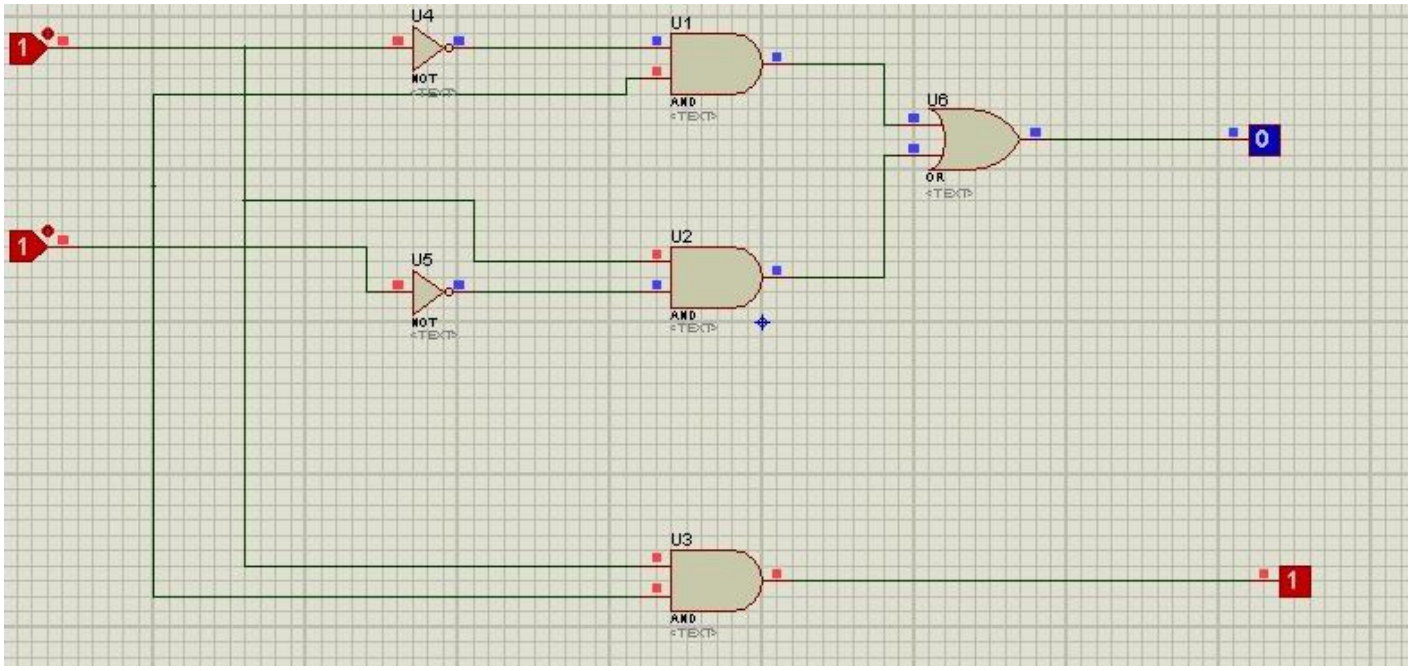


PROTEUS Simulation:




0
1
U4
NOT
U1
AND
U6
OR
1
U2
U5
NOT
AND
U3
AND
0


1
0
U4
NOT
U1
AND
U6
OR
1
U2
U5
NOT
AND
U3
AND
0



Lab Task 3: (2 marks)

Label each gate output in the above circuit and derive algebraic expressions for SUM and Carry Out. Fill in the following truth table and determine the function performed by the circuit.

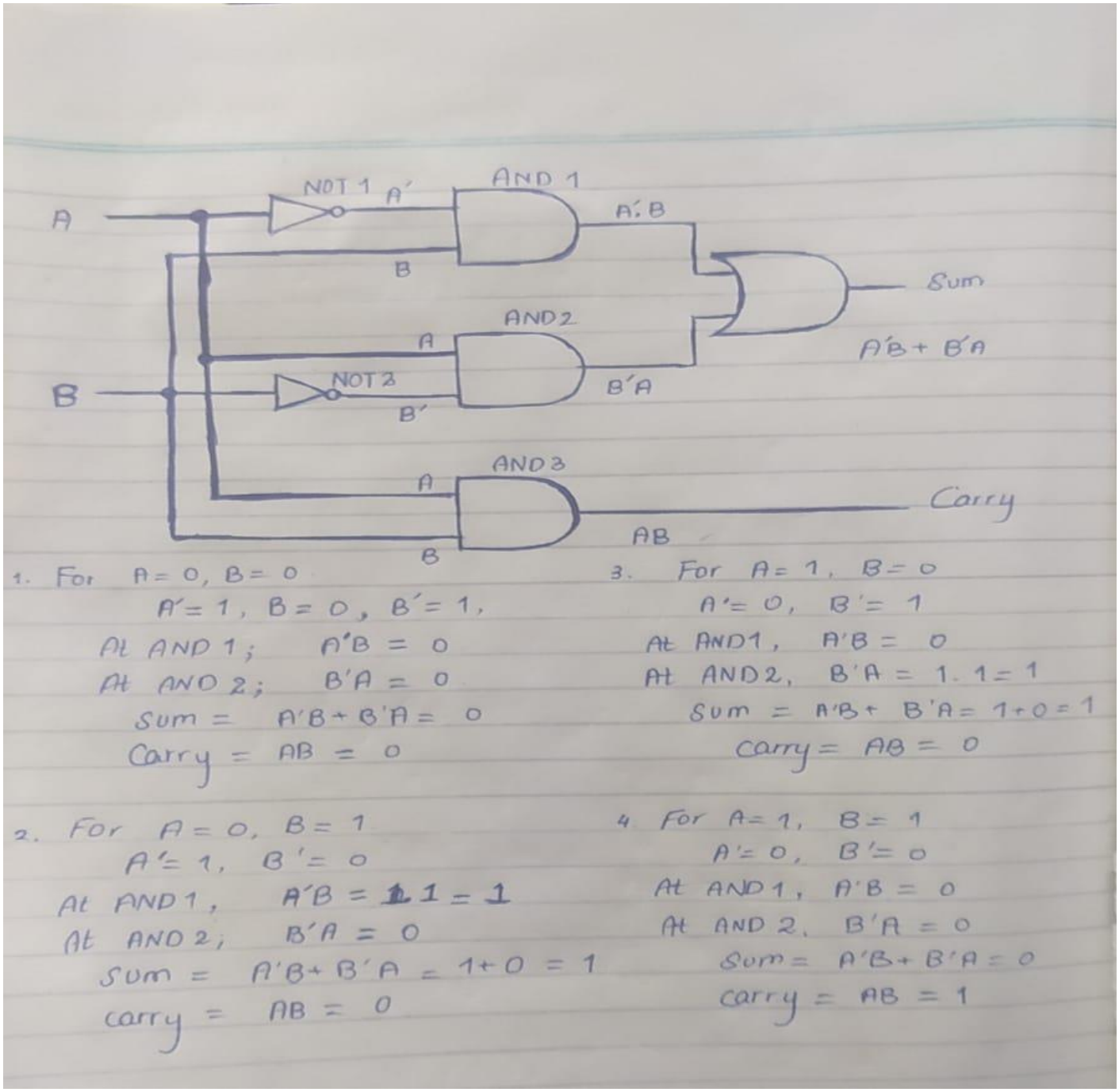
Truth Table:

x	y	Sum	Carry Out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Function:

This circuit is adding two bits and calculating the sum and carry out. Thus, it works as a 2-bit adder circuit.

Derivation:


A
B
NOT 1
A'
AND 1
A'B
B
AND2
A
NOT 2
B'
B'A
Sum
A'B + B'A
AND 3
A
B
AB
Carry
1. For A = 0, B = 0
A' = 1, B = 0, B' = 1,
At AND 1; A'B = 0
At AND 2; B'A = 0
Sum = A'B + B'A = 0
Carry = AB = 0
2. For A = 0, B = 1
A' = 1, B' = 0
At AND 1, A'B = 1.1 = 1
At AND 2, B'A = 0
Sum = A'B + B'A = 1 + 0 = 1
carry = AB = 0
3. For A = 1, B = 0
A' = 0, B' = 1
At AND1, A'B = 0
At AND2, B'A = 1.1 = 1
Sum = A'B + B'A = 1 + 0 = 1
carry = AB = 0
4. For A = 1, B = 1
A' = 0, B' = 0
At AND 1, A'B = 0
At AND 2, B'A = 0
Sum = A'B + B'A = 0
carry = AB = 1

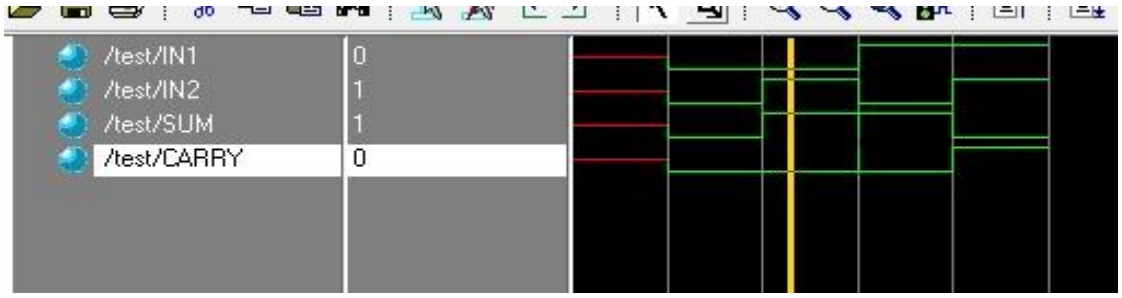
Lab Task 4: (2 marks)

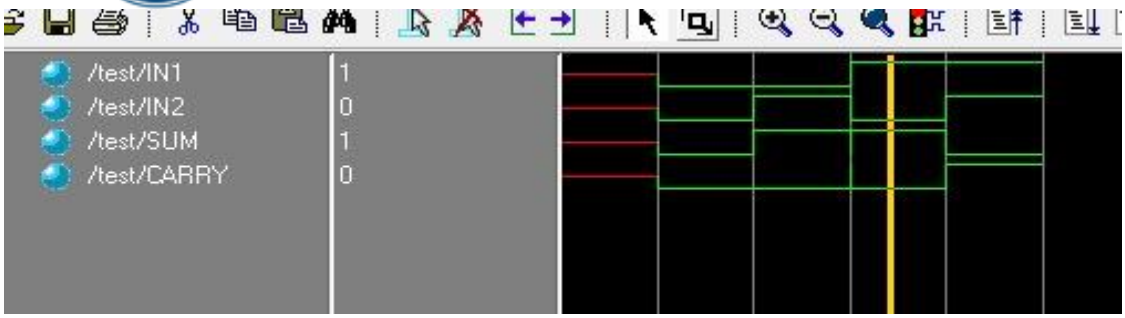
After determining the function performed by the circuit given in Lab Task 2, write the Verilog description of the circuit in **dataflow**. Comment on the two different modeling levels you used to model the same circuit. (Paste snapshots of the codes and stimuluses below)

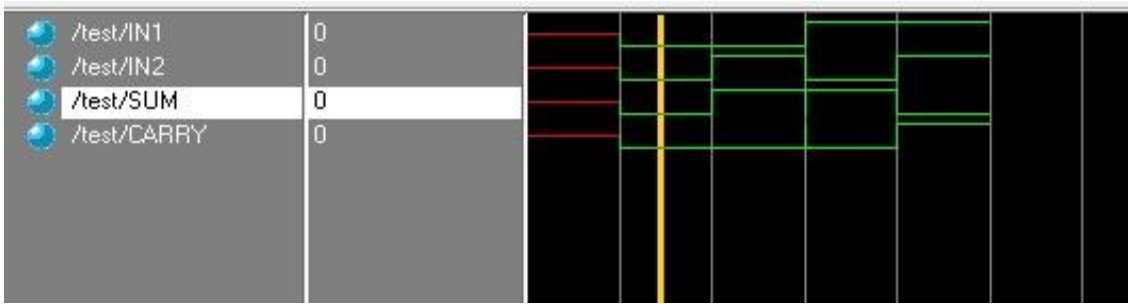
Dataflow Code:

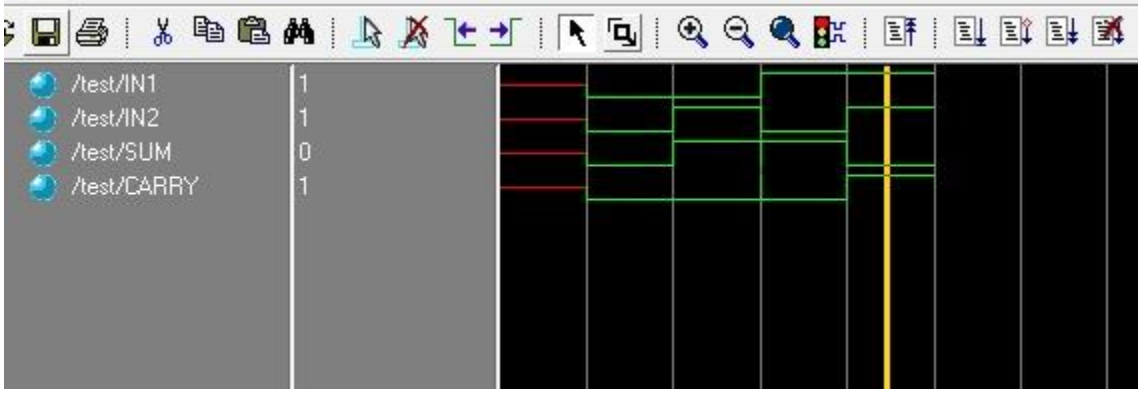
```
 1 module circuit (sum, carry, in1,in2);
 2 input in1,in2;
 3 output sum,carry;
 4 assign sum= ((~in1) & in2) | ( (~in2) & in1);
 5 assign carry = in1 & in2;
 6 endmodule
 7
 8 module test;
 9 reg IN1,IN2;
10 wire SUM, CARRY;
11 circuit test1(SUM, CARRY, IN1,IN2);
12 initial
13 begin
14         #100 IN1=1'b0;IN2=1'b0;
15         #100 IN1=1'b0;IN2=1'b1;
16         #100 IN1=1'b1;IN2=1'b0;
17         #100 IN1=1'b1;IN2=1'b1;
18 end
19 endmodule
20
```

Simulation Results:









Observations/Comments:

In this lab, we got acquainted with Verilog HDL and used two types of abstraction: Gate-Level modelling and Dataflow Modelling. Dataflow modelling has much easier syntax, which is also shorter in length. This is because the gate level modelling requires us to define not only the inputs and outputs, but the output of every single gate individually which can serve as an input to the next gate. This is done using defining wires. On the contrary, in dataflow modelling, one can simply assign an output from a gate an algebraic expression in terms of the original inputs which eliminates the need for using the declarations of wires.