North South University

Department of Electrical & Computer Engineering

Lab Report

Experiment No:	01
Experiment Title:	Digital Logic and Boolean Functions
Course Code:	**CSE231L**
Section:	**10**
Course Name:	**Digital Logic Design Lab**
Lab Group #:	07
Written By:	Tawhid Hasan (ID-2413054042)
Date of Experiment:	27/09/2025
Date of Submission:	04/10/2025

Group Members ID:	**Group Members Name:**
2413054042	Tawhid Hasan
2413692042	Tasvirul Hasan Riyad
2411600042	Md Tayasshuk Imam
2322723642	Md Mafiul Haque

Lab-1: Digital Logic and Boolean Functions

Objective: The main goal of the lab was to understand the fundamental principles of the basic logic gates and hands-on experience with the gates we usually only study in theory, including AND, OR, NOT, NAND, NOR, XOR. We also aimed to understand how Boolean functions can be reppresented using truth table, logic diagrams and Boolean Algebra and also to prove the extension of inputs of AND and OR gates using Associate law.

List of Equipment:

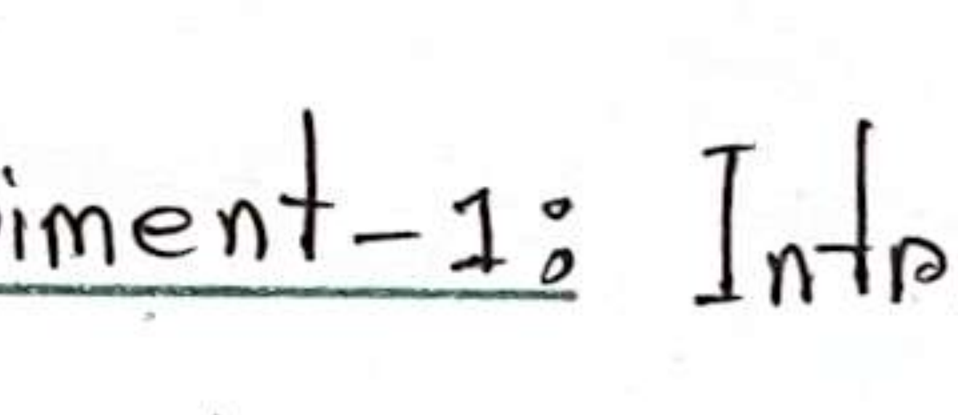
1. Trainer Board.
2. Connecting Wires.
3. ICs:
   - 7408 Quadruple 2-input AND gate.
   - 7432 Ouadruple 2-input OR gate.
   - 7404 Hex Inverter (NOT gate)
   - 7400 Quadruple 2-input NAND gate
   - 7402 Quadruple 2-input NOR gate
   - 7486 Quadruple 2-input XOR gate.

Theory: Logic gates are basic building block in digital electronic. By performing a specific operation on binary inputs, it produces a single output. The behaviour of these gates is defined by their truth tables, which shows the output for every possible combination of inputs. Boolean algebra provides a mathematical way for analyzing and simplifying the logic circuits. Combinational logic circuits are that type of circuit where the output depends on only current inputs.

Experiment-1: Introduction to Basic Logic Gates.

AND Gate: A logical gate that produces a HIGH output only when all of the inputs are HIGH.

↳ Symbol:



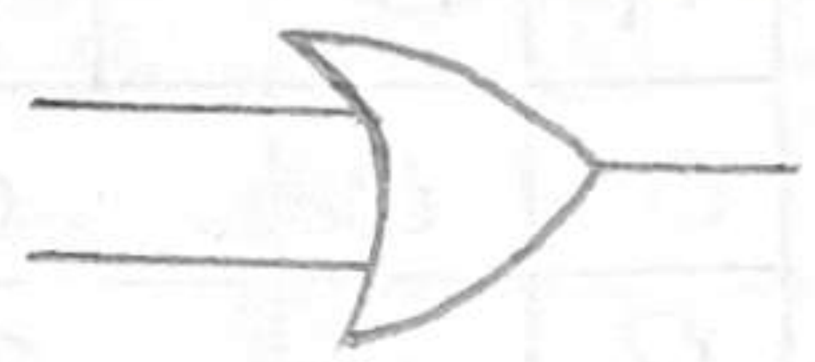
↳ Truth Table:

A	B	AND (A.B)
0	0	0
0	1	0
1	0	0
1	1	1

IC: 7408 Quadruple 2-input AND Gate.

OR Gate: This logic gate produces a HIGH output when one or more inputs are HIGH.

↳ Symbol:

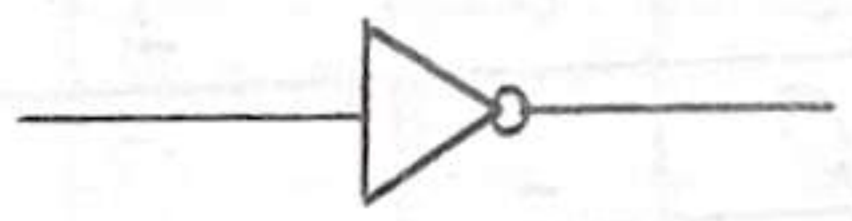


↳ Truth Table:

A	B	OR (A+B)
0	0	0
0	1	1
1	0	1
1	1	1

NOT Gate: This logic gate inverts its input.

↳ Symbol:

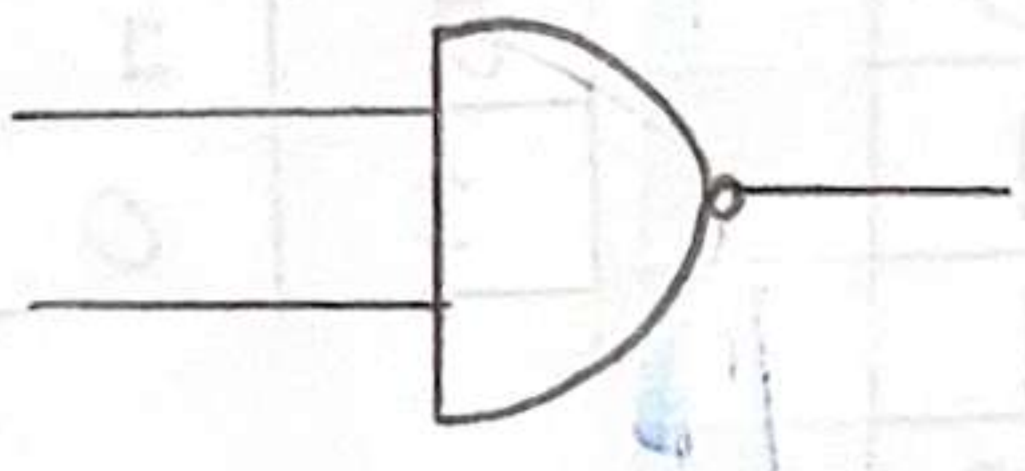


↳ Truth Table:

A	NOT $\bar{A}$
0	1
1	0

NAND Gate:

↳ Symbol:

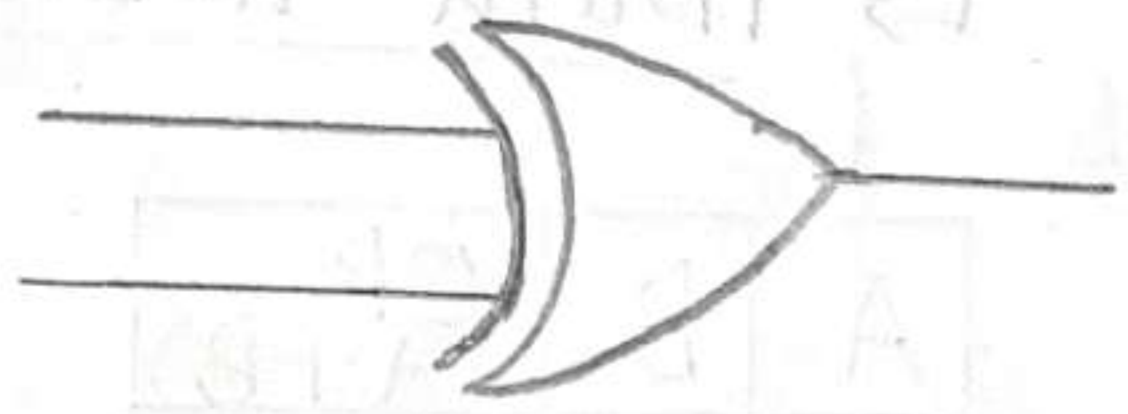


↳ Truth Table:

A	B	NAND $F=\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

XOR Gate:

↳ Symbol:

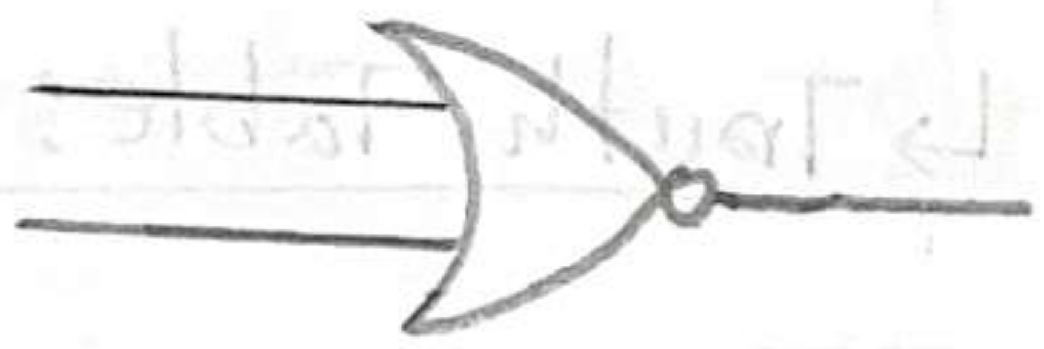


↳ Truth Table:

A	B	XOR $F = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

NOR Gate:

↳ Symbol:



↳ Truth Table:

A	B	NOR $F = \overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

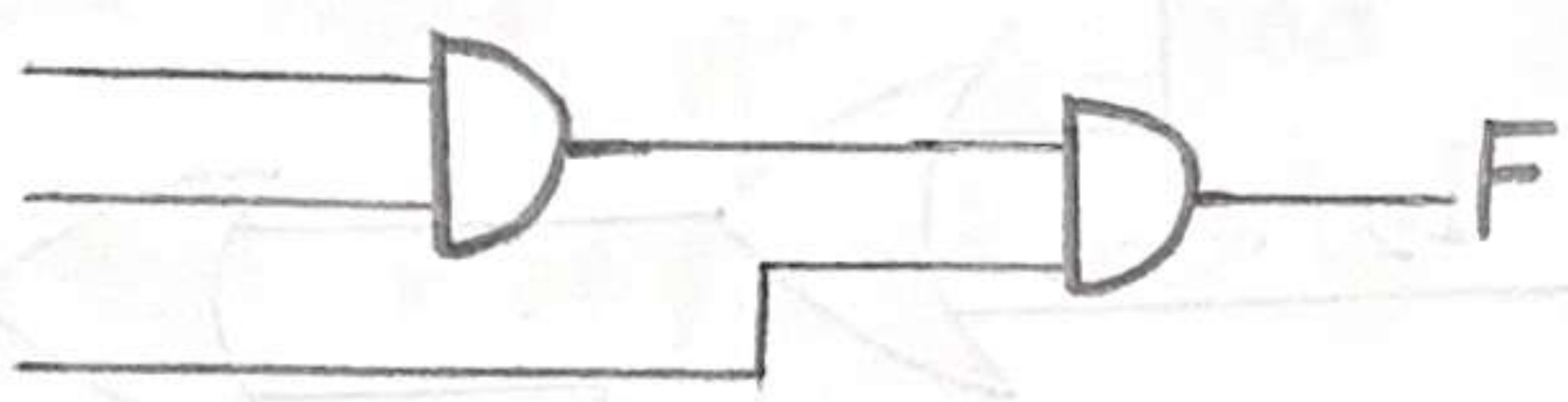
#Experimental Data Table:

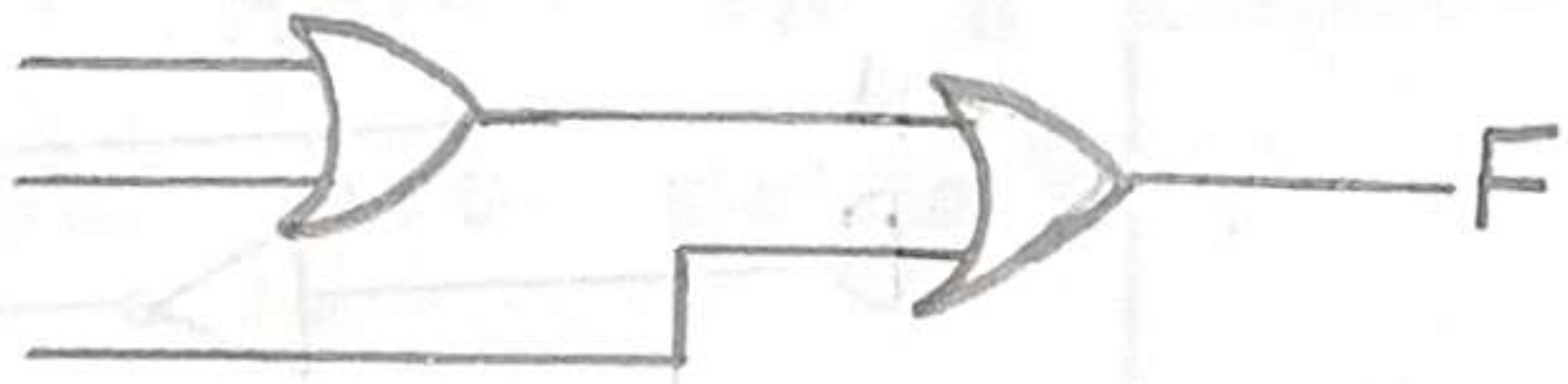
Input	AND	OR	NOR	NAND	XOR
00	0	0	1	1	0
01	0	1	0	1	1
10	0	1	0	1	1
11	1	1	0	0	0

Input	NOT
0	1
1	0

Experiment-2: Constructing 3-input AND & OR Gates from 2-input AND & OR gates.

circuit diagram:



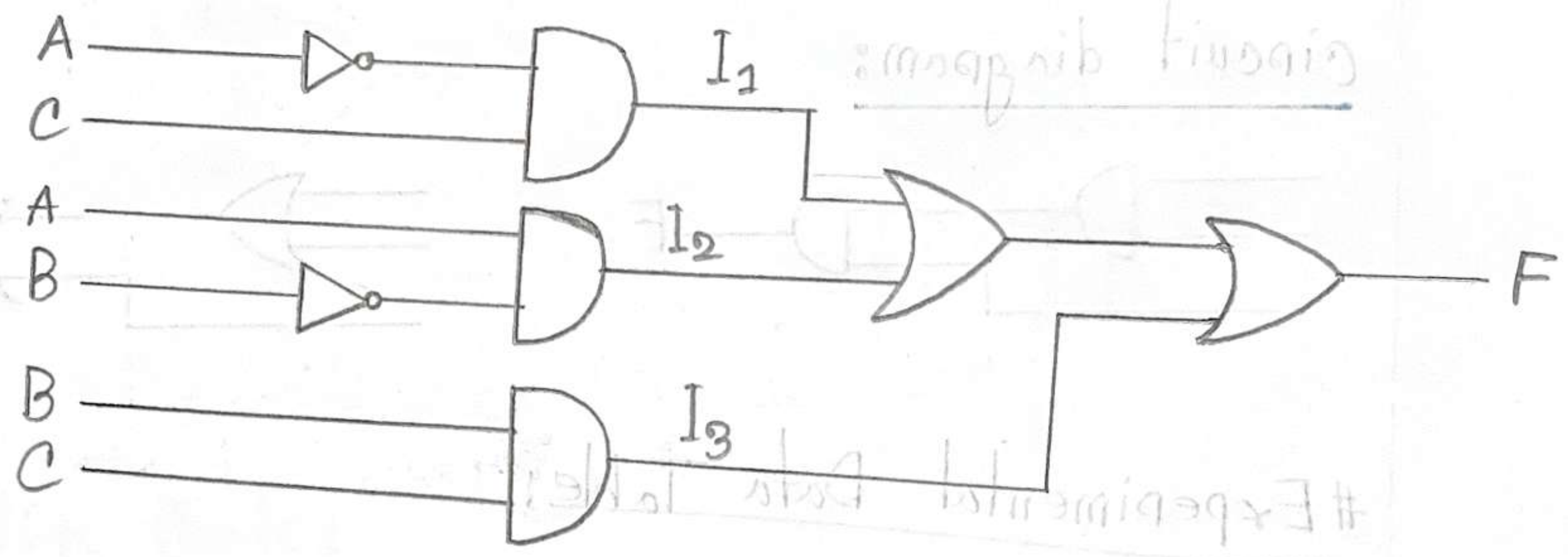


#Experimental Data Table:

ABC	F=ABC	F=A+B+C
000	0	0
001	0	1
010	0	1
011	0	1
100	0	1
101	0	1
110	0	1
111	1	1

Experiment-3: Implementation of Boolean Functions.

Circuit diagram:



Experimental Data Table:

A B C	$I_1 = A'C$	$I_2 = AB'$	$I_3 = BC$	$F = I_1 + I_2 + I_3$
0 0 0	0	0	0	0
0 0 1	1	0	0	1
0 1 0	0	0	0	0
0 1 1	1	0	1	1
1 0 0	0	1	0	1
1 0 1	0	1	0	1
1 1 0	0	0	0	0
1 1 1	0	0	1	1

#Question and Answers:

1. What are the names of the ICs that you would need if you wanted to use 13 AND gates 12 NOT gates and 15 NOR gates in a circuit? How many of each IC would you need?

Answer:

AND gates: IC 7408, for 13 gates $\lceil 13/4 \rceil = 4$ ICs

NOT gates: IC 7404, for 12 gates $\lceil 12/6 \rceil = 2$ ICs

NOR gates: IC 7402, for 15 gates $\lceil 15/4 \rceil = 4$ ICs

2. How can you power your logic ICs if the +5V port of your trainer board stops working?

Answer: If the +5V port of my trainer board stops working, I will power the logic ICs from the trainer input toggle switches.

3. Explain the Associative Law of Boolean algebra.

Answer: The Associative Law of Boolean algebra states that, when performing the same operation (AND or OR) on more than two variables, the order of grouping the variables does not change the final result.

↳ For AND: $A.(B.C) = (A.B).C$

↳ For OR: $A+(B+C) = (A+B)+C$

4. What is truth table? Draw the truth table for XNOR gate.

Answer:

A Truth table is a chart that lists all possible input combinations of a digital circuits and shows the corresponding output for each one.

A	B	XNOR F
0	0	1
0	1	0
1	0	0
1	1	1

Here ~~is~~ in the right side this is a Truth table for XNOR gate.

5. Let us assume you have two logical inputs A and B. If you pass A and B through a NAND gate and then pass the output of the NAND gate through a NOT gate, what logical operation will your final output represent? What is the name of the Boolean Algebra theorem that can be used to find this answer?

Answer:

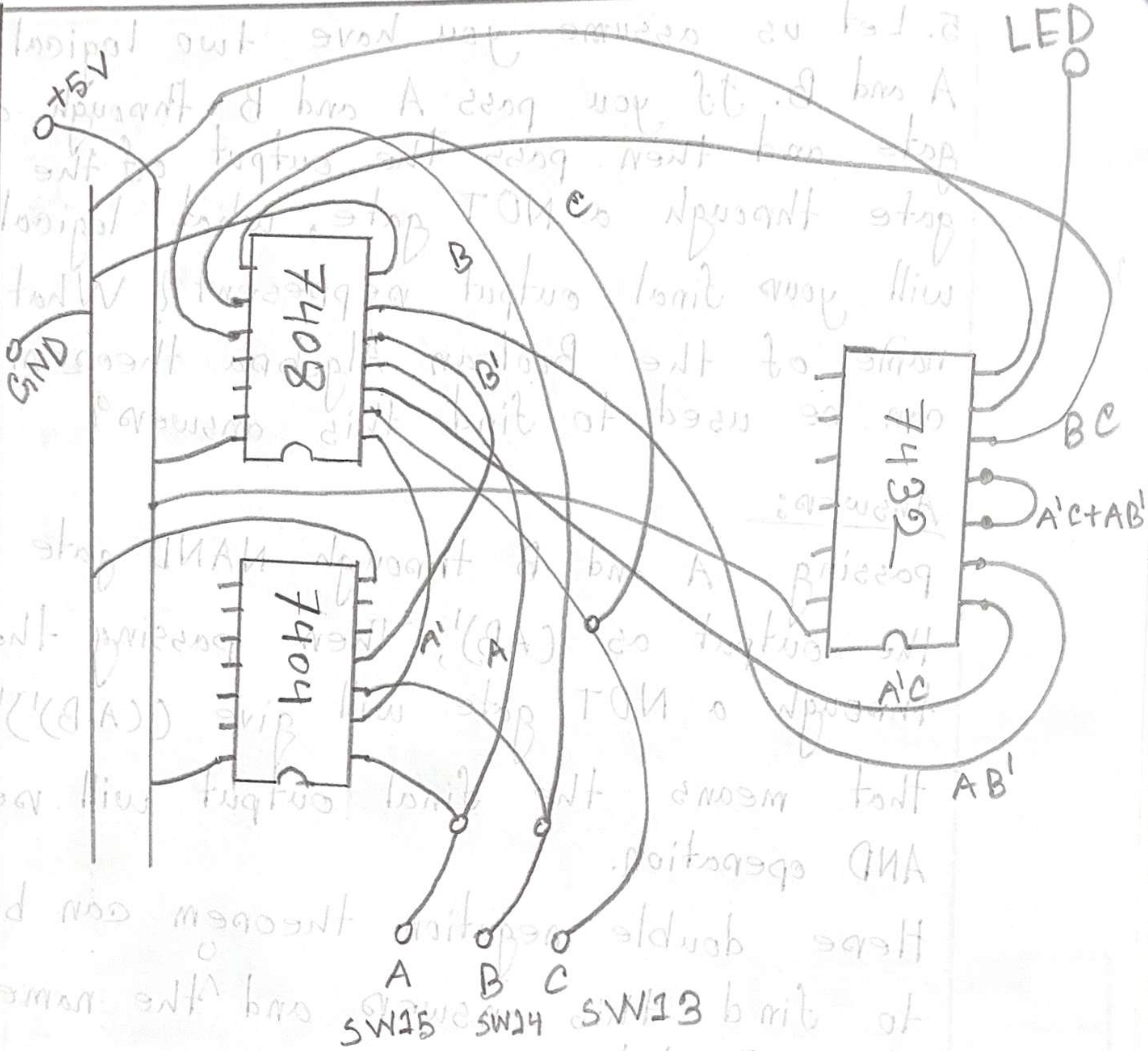
passing A and B through NAND gate will give the output as $(AB)'$, Then passing the output through a NOT gate will give $((A.B)')' = (A.B)$ that means the final output will represent AND operation.

Here double negation theorem can be used to find this answer and the name of it is Involution.

6. Draw the IC Diagram for the circuit in Figure F. 3.1. In place of the logic gates, draw the ICs and all the connections required to make the circuit work.

Answer:

(P.T.O)



7. How can you use a 3 input AND gate? Can you use the same method to use 3- input OR gate as a 2-input OR gate?

Answer: To use a 3-input AND gate as a 2-input AND gate, we have to connect the unused third input to a permanent logic HIGH (1). Since A.B.1 = A.B, the output depends only on A and B.

Not the same but similar method will ~~be~~ work for a 3-input OR gate. Here the third input should be connected to a permanent logic LOW (0), Since $A+B+0 = A+B$, the output depends only on A and B.

Discussion:

During the lab session, we first tested each basic logic gate IC. From the first test run our ICs were working perfectly. The LED outputs for the AND, OR and NOT gates matched the theoritical truth tables perfectly. But for the NAND, NOR and XOR everything were not perfect. In our second attempt using NAND IC we got the perfect result.

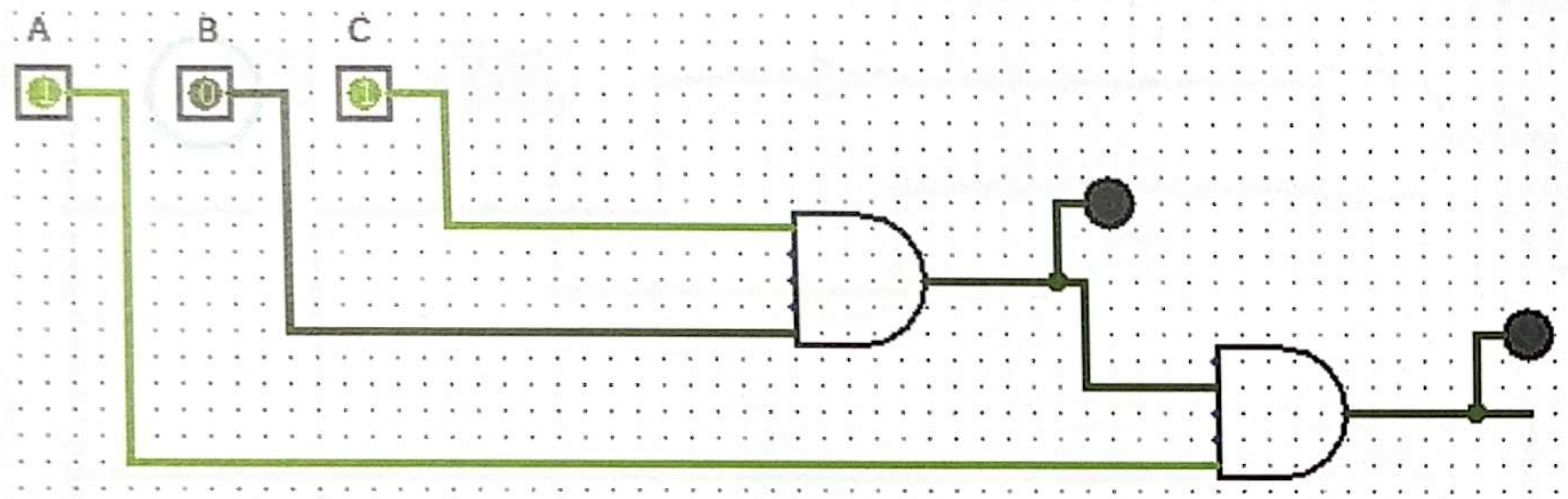
After the first experiment, we build a circuit of 3-input gates, the practical result confirmed the associative Law. (P.T.O)

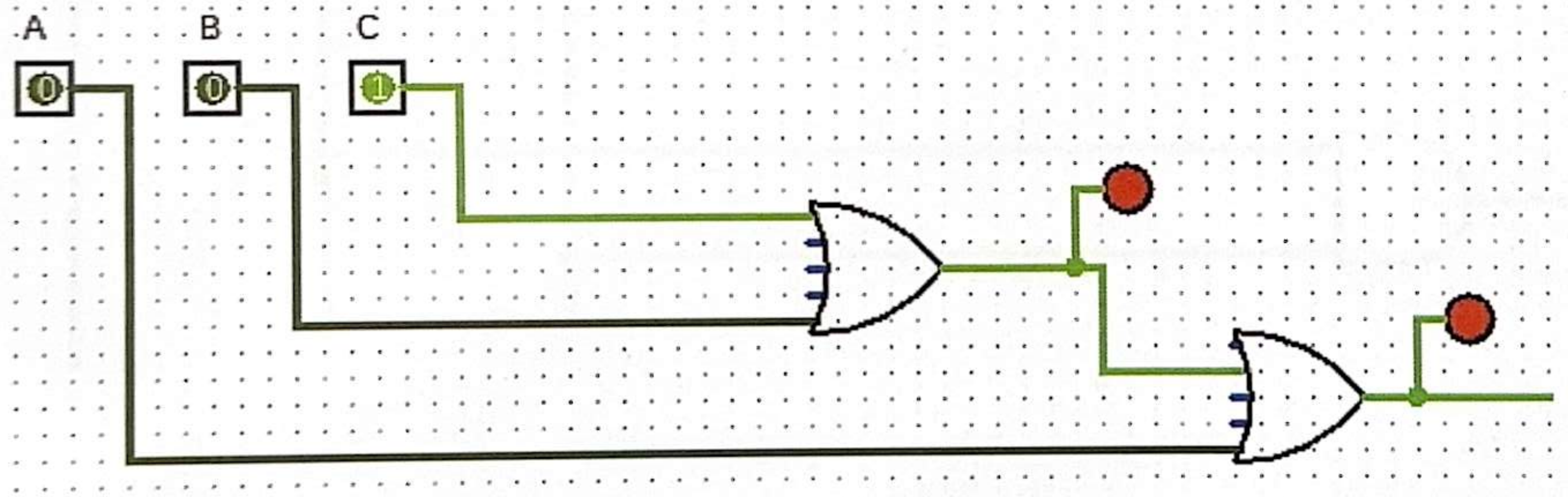
We saw that the LED for the 3-input AND gate only lit up when all the three ~~g~~ switches were ON.

For the final and third experiment, the implementation of $F = A'C + AB' + BC$ was the most complex part of the lab session. We faced some difficulties in building the circuit. At our first attempt the result was different from the truth table. Then we again checked all the connection and then everything worked perfectly. ~~and w~~ The result matched with the truth table.

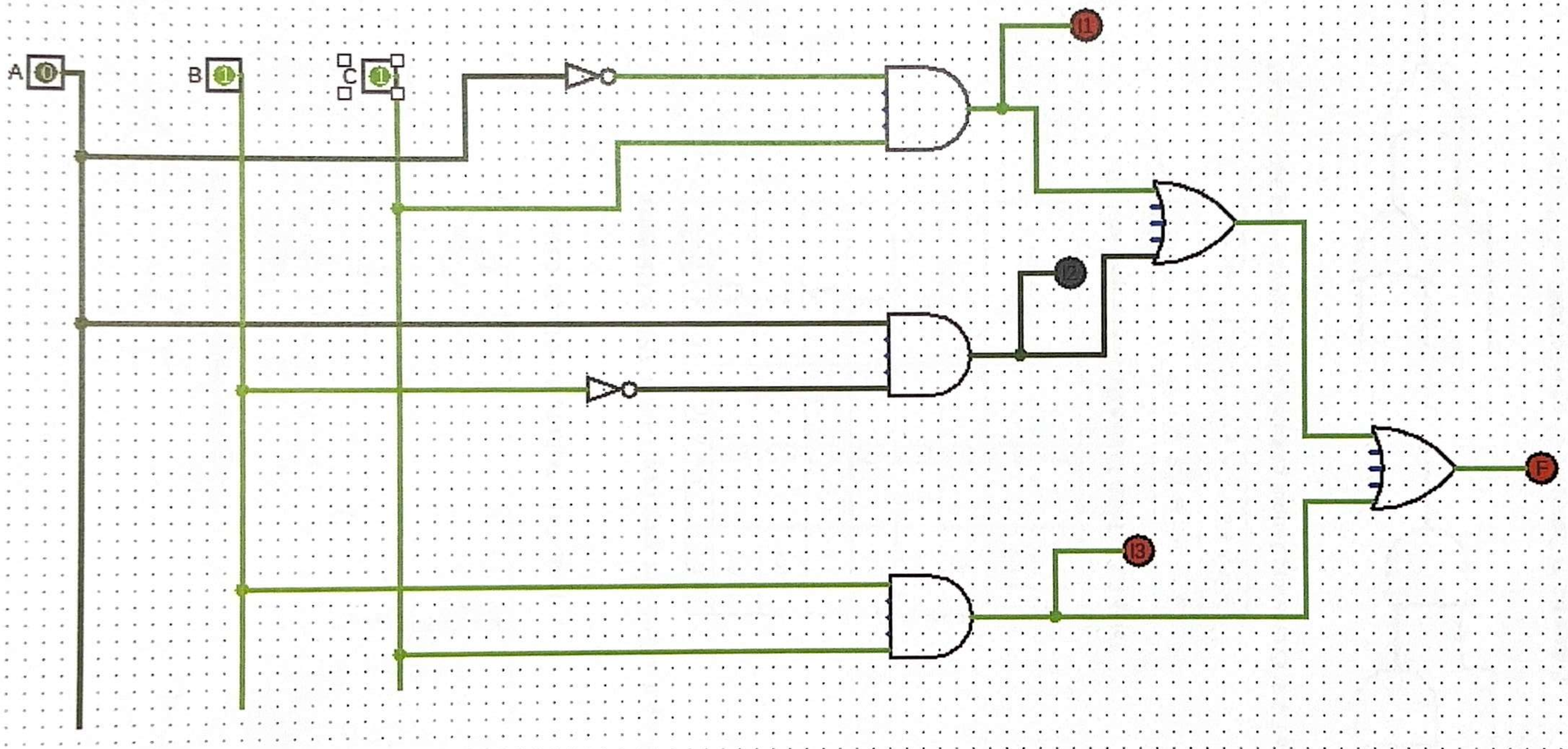
Overall, the lab session was very helpful to learn and get hands-on experience on concept of Boolean algebra, truth table, ICs, and working circuits.

Tawhid Hasan
ID:2413054042





Tawhid Hasan
ID:2413054042



F. Data Sheet

F.1 Introduction to Basic Logic Gates

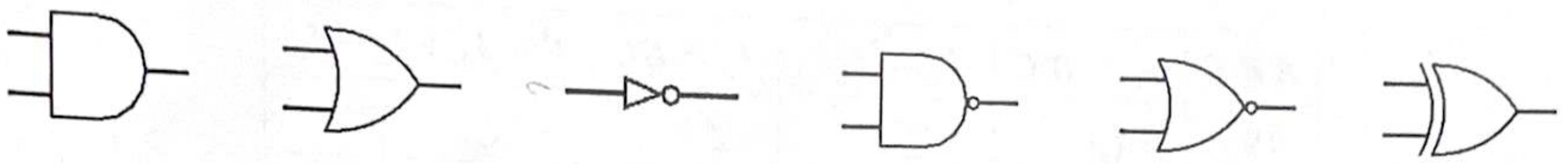


Figure F.1.1: Pin configurations of gates in ICs

Input $A\ B$	AND $F = A \cdot B$	OR $F = A + B$	NAND $F = \overline{A \cdot B}$	XOR $F = A \oplus B$	NOR $F = \overline{A + B}$
0 0	0	0	1	0	1
0 1	0	1	1	1	0
1 0	0	1	1	1	0
1 1	1	1	0	0	0

Input A	NOT $F = \overline{A}$
0	1
1	0

Table F.1.1: Truth Table of Logic Gates

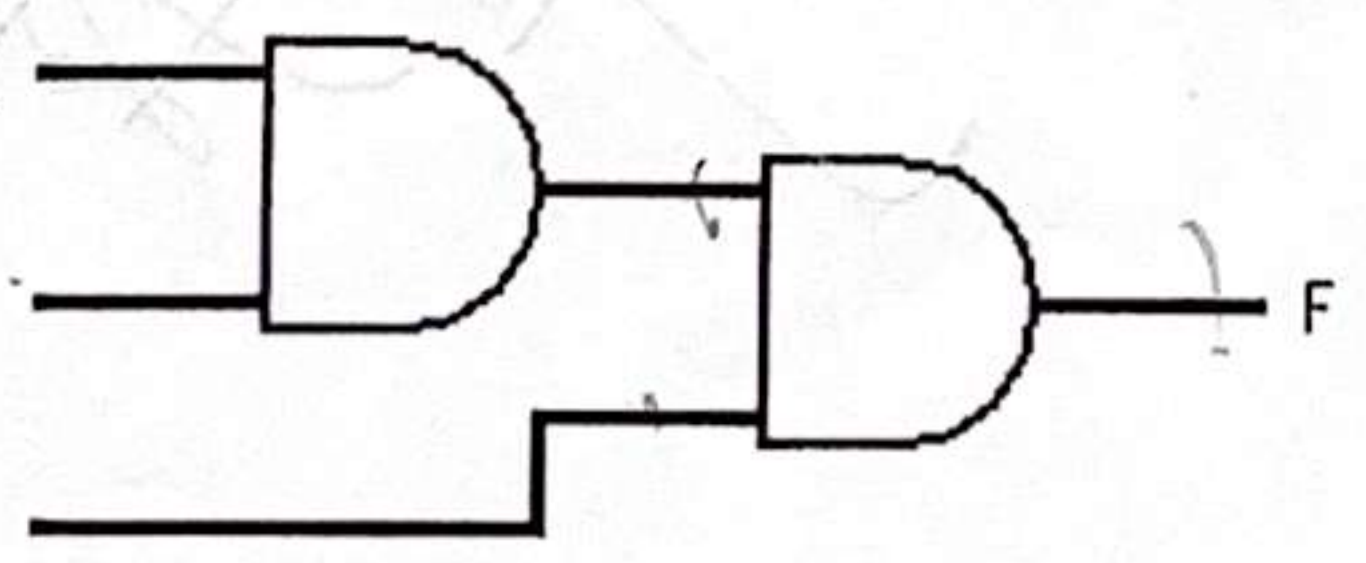
F.2 Constructing 3-input AND & OR gates from 2-input AND & OR gates

$A\ B\ C$	$F = ABC$	$F = A + B + C$
0 0 0	0	0
0 0 1	0	1
0 1 0	0	1
0 1 1	0	1
1 0 0	0	1
1 0 1	0	1
1 1 0	0	1
1 1 1	1	1

Table F.2.1: Truth Tables for 3-input AND and OR

$F = ABC =$	$(A \cdot B) \cdot C$
$F = A + B + C =$	$(A + B) + C$

Table F.2.2: Expressing 3-input gates as 2-input gates using associative law.



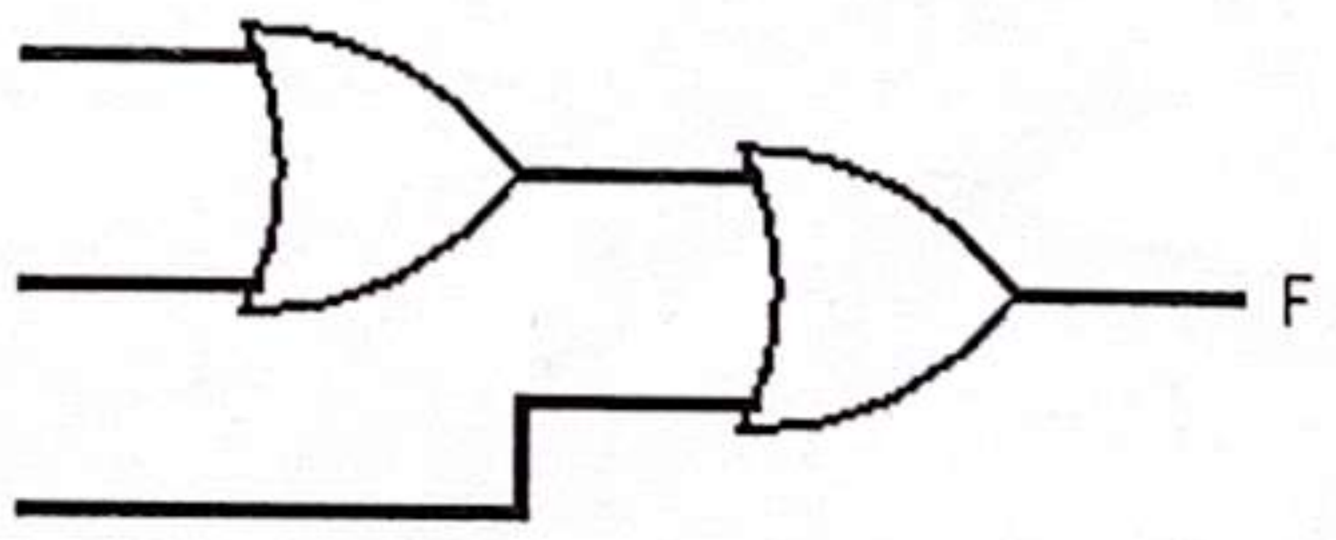


Figure F.2.1: Extension of inputs of AND and OR gates

F.3 Implementation of Boolean Functions

ABC	$I_1 = A'C$	$I_2 = AB'$	$I_3 = BC$	$F = I_1 + I_2 + I_3$
000	0	0	0	0
001	1	0	0	1
010	0	0	0	0
011	1	0	1	1
100	0	1	0	1
101	0	1	0	1
110	0	0	0	0
111	0	0	1	1

Table F.3.1: Truth Table for the given Boolean Function

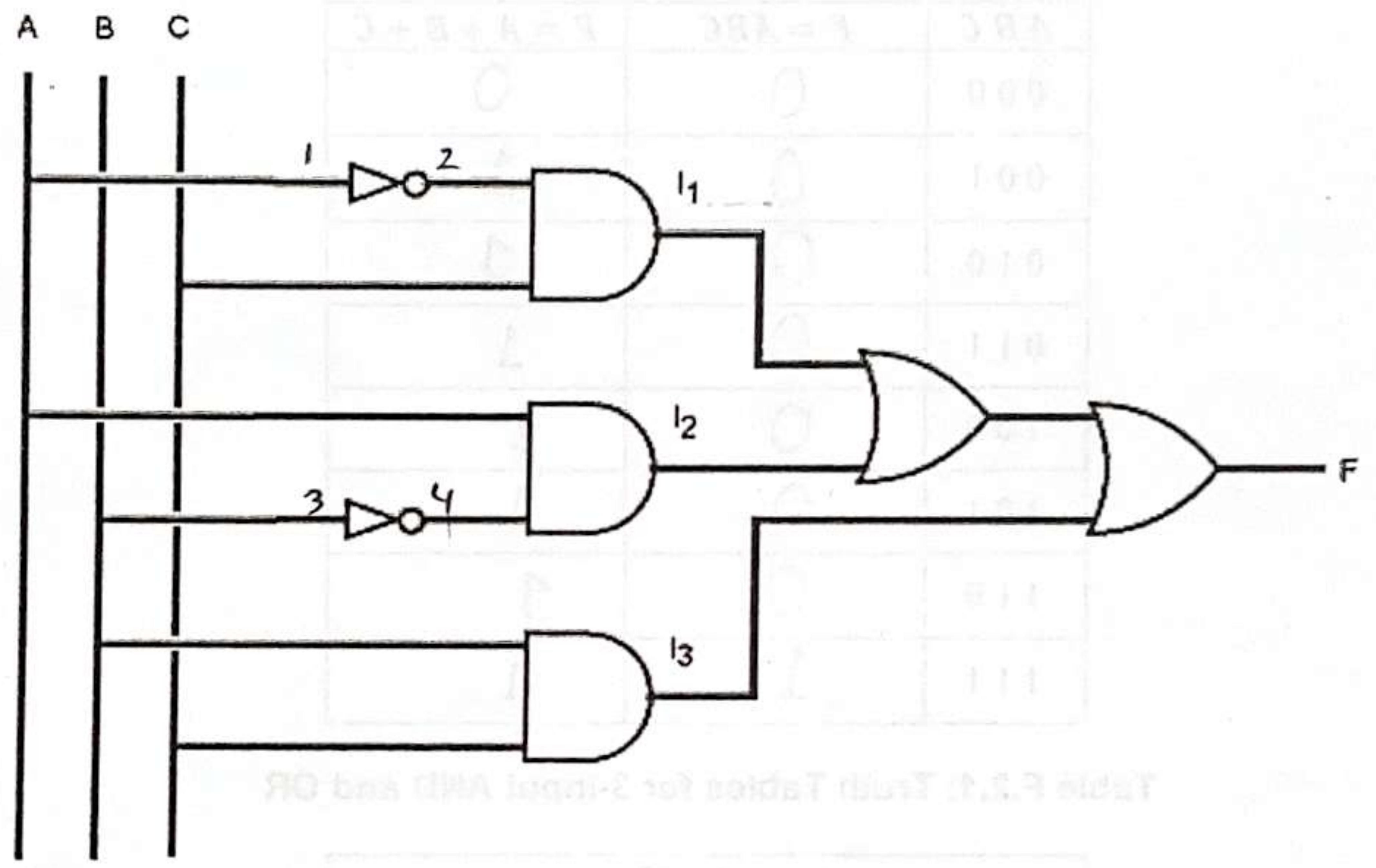


Figure F.3.1: Logic Diagram for the given Boolean Function

100%

[illegible]