Course Name:	**Digital Design Laboratory**	**Semester:**	**III**
Date of Performance:	**15 / 07 / 2024**	**Batch No:**	**D3**
Faculty Name:	**Mansi Kambli**	**Roll No:**	**16010123294**
Faculty Sign & Date:		**Grade/Marks:**	**___/25**

Experiment No: 1
Title: Study of Basic Gates and Universal Gates

Aim and Objective of the Experiment:
Understand Basic Logic Gates and Universal Gates

COs to be achieved:
CO1: Recall basic gates & logic families and binary, octal & hexadecimal calculations and conversions.

Tools used:
Trainer kits

Theory:

Logic gates are electronic circuits that perform logical operations on one or more input signals to produce an output signal based on a set of logical rules. Logic gates can be classified into the following categories:

1. Basic Gates:
   a. AND Gate: The AND gate produces a high output (1) only when all of its inputs are high (1).
   b. OR Gate: The OR gate produces a high output (1) if any of its inputs is high (1).
   c. NOT Gate (Inverter): The NOT gate produces the logical complement of its input. It takes a single input and produces the opposite value as the output.
2. Derived Gates:
   a. NAND Gate: The NAND gate is a combination of an AND gate followed by a NOT gate. It produces the inverse of the AND gate's output. It outputs a low (0) only when all of its inputs are high (1).
   b. NOR Gate: The NOR gate is a combination of an OR gate followed by a NOT gate. It produces the inverse of the OR gate's output. It outputs a high (1) only when all of its inputs are low (0).
   c. XOR Gate (Exclusive OR): The XOR gate produces a high output (1) when the number of high inputs is odd. It outputs a low (0) when the number of high inputs is even.
   d. XNOR Gate (Exclusive NOR): The XNOR gate produces a high output (1) when the number of high inputs is even. It outputs a low (0) when the number of high inputs is odd.
3. Universal Gates:

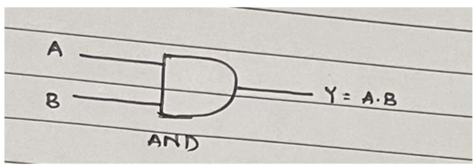
   NAND and NOR gates are considered universal gates because any logic function can be implemented using only NAND gates or only NOR gates. This means that with a sufficient number of NAND or NOR gates, you can create circuits that can perform any logical operation.

Implementation Details

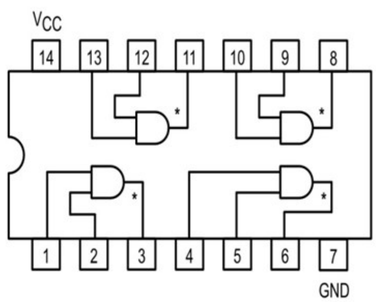
1. AND Gate:

   Y = A.B

Symbol :



Pin Diagram



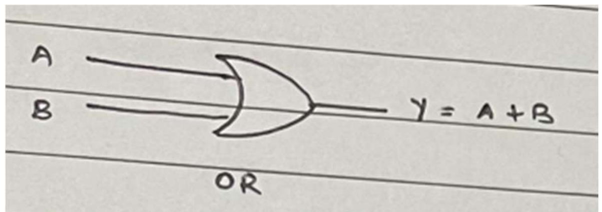
Truth Table:

A	B	OUTPUT
0	0	0
0	1	0
1	0	0
1	1	1

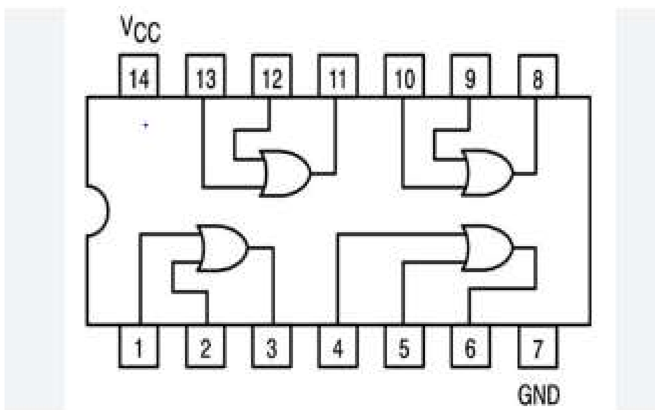
‘

2. OR Gate: Y =A+B

Symbol



Pin Diagram

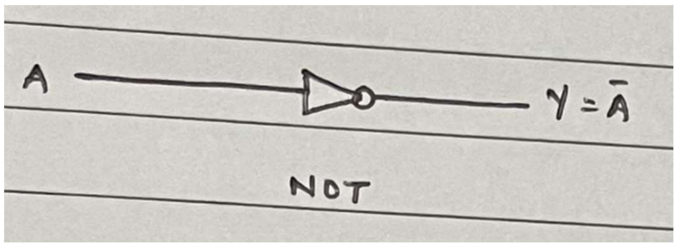


Truth Table:

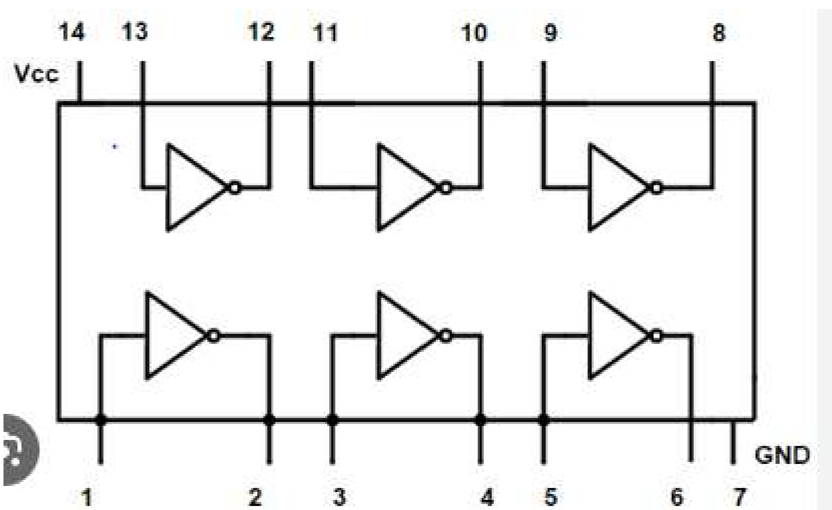
A	B	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	1

3. NOT Gate: $Y = \bar{A}$

Symbol



Pin Diagram

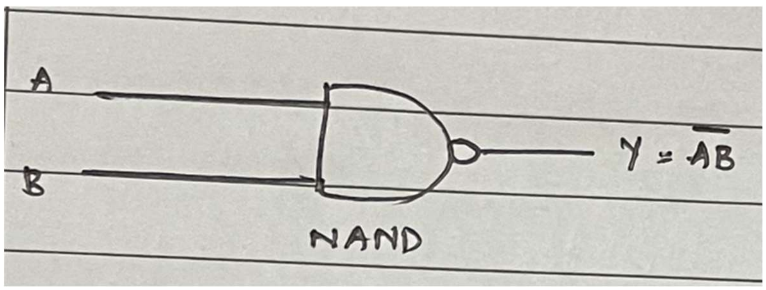


Truth Table:

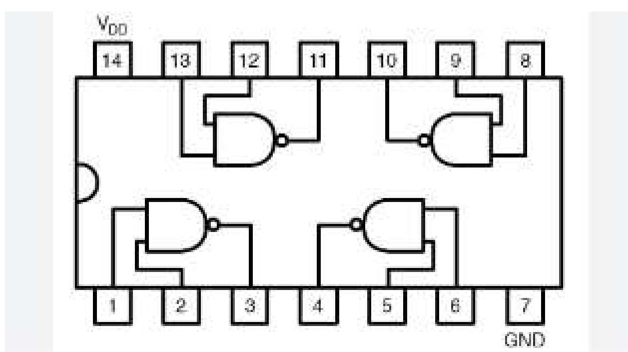
A	Output
0	1
1	0

4. NAND Gate: $Y = (A.B)^c$

Symbol



Pin Diagram

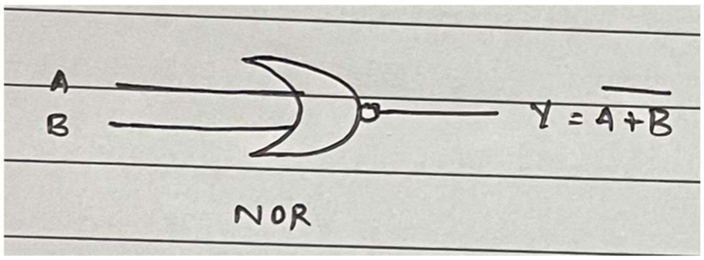


Truth Table:

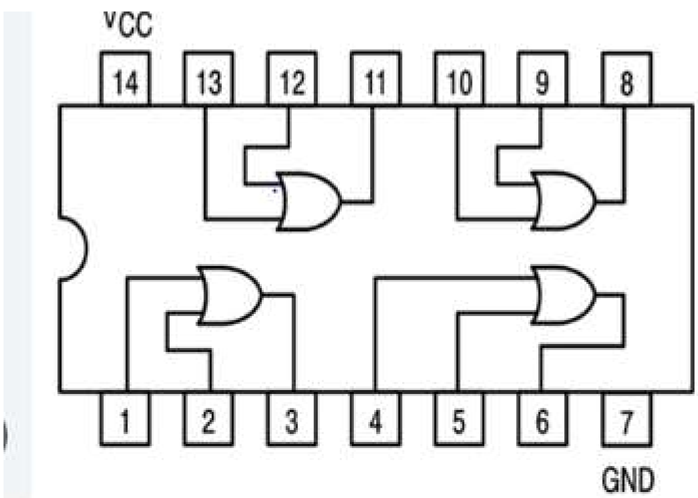
A	B	OUTPUT
0	0	1
0	1	1
1	0	1
1	1	0

5. NOR Gate: Y =$(A+B)^c$

Symbol



Pin Diagram

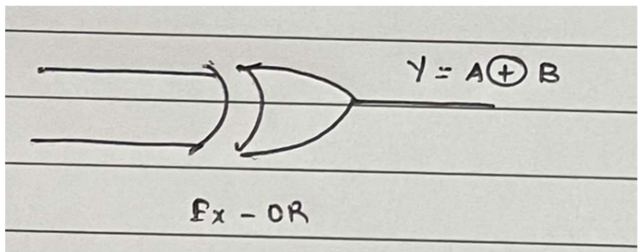


Truth Table:

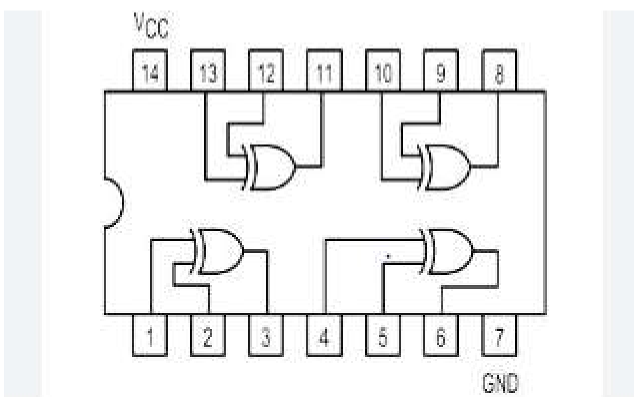
A	B	OUTPUT
0	0	1
0	1	0
1	0	0
1	1	0

6. XOR Gate: Y =A $\oplus$ B

Symbol



Pin Diagram

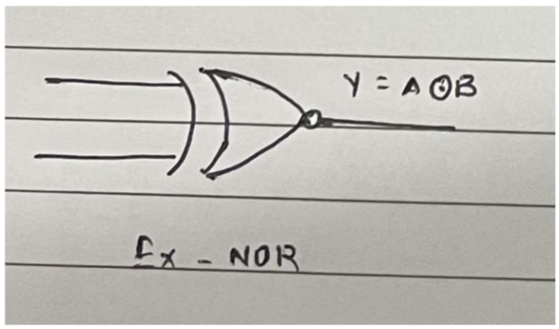


Truth Table:

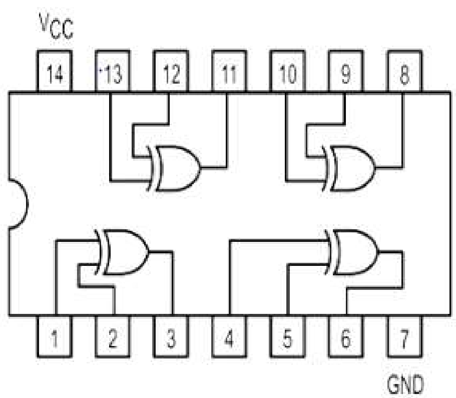
A	B	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	0

7. XNOR Gate: $Y = (A \oplus B)^c$

Symbol



Pin Diagram

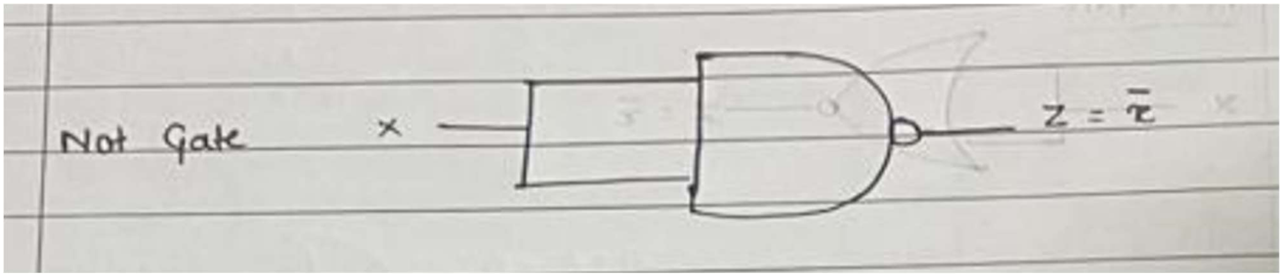


Truth Table:

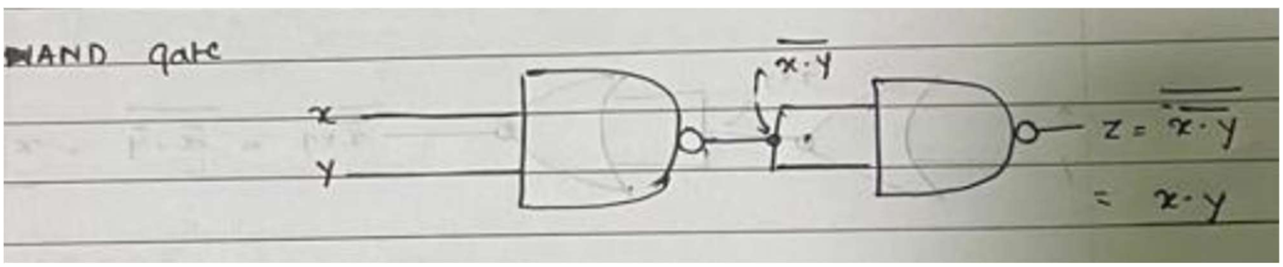
A	B	OUTPUT
0	0	1
0	1	0
1	0	0
1	1	1

Implementation Using NAND Gate

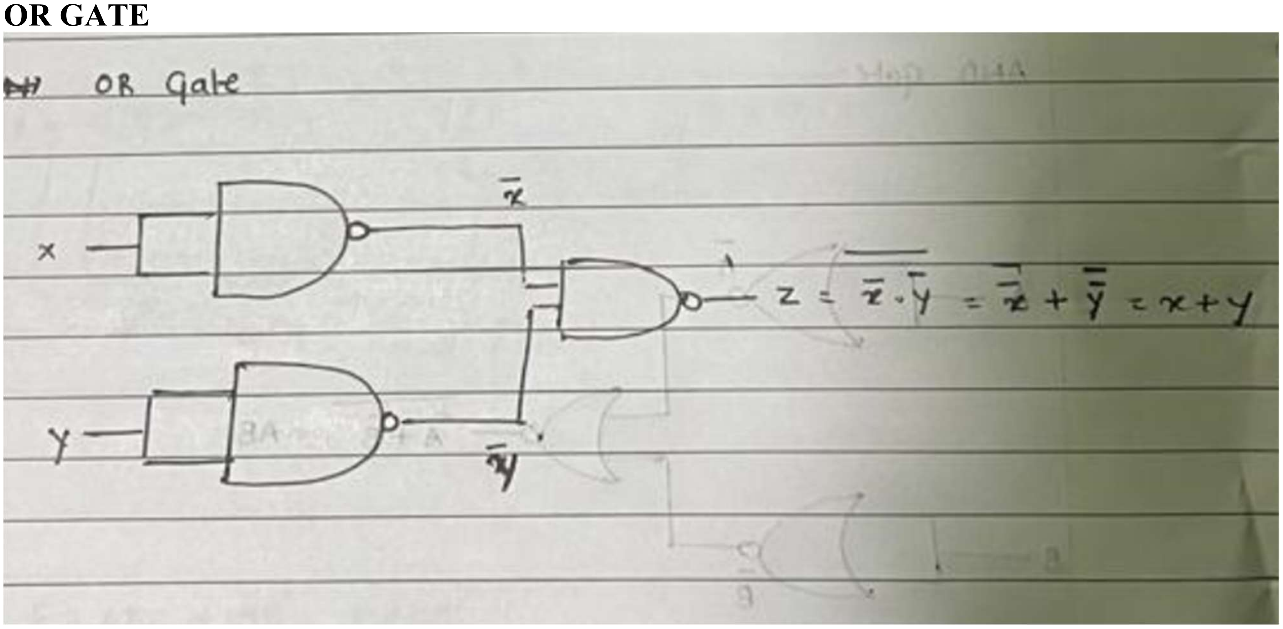
NOT GATE



AND GATE

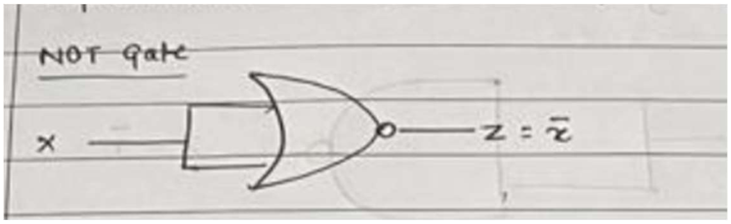


OR GATE

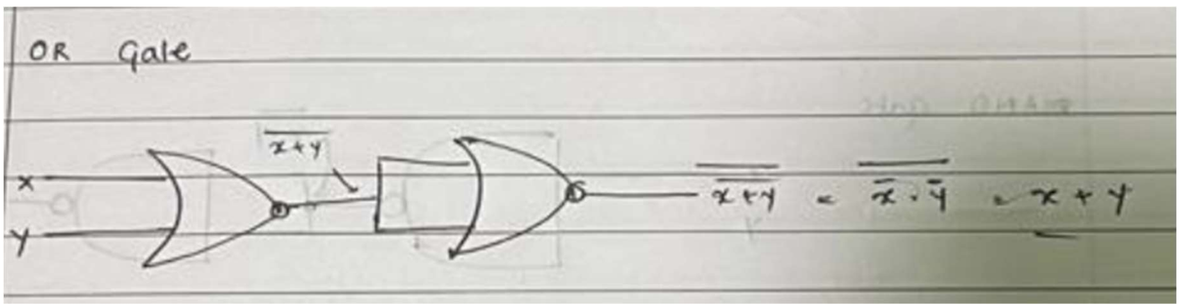


Implementation Using NOR Gate

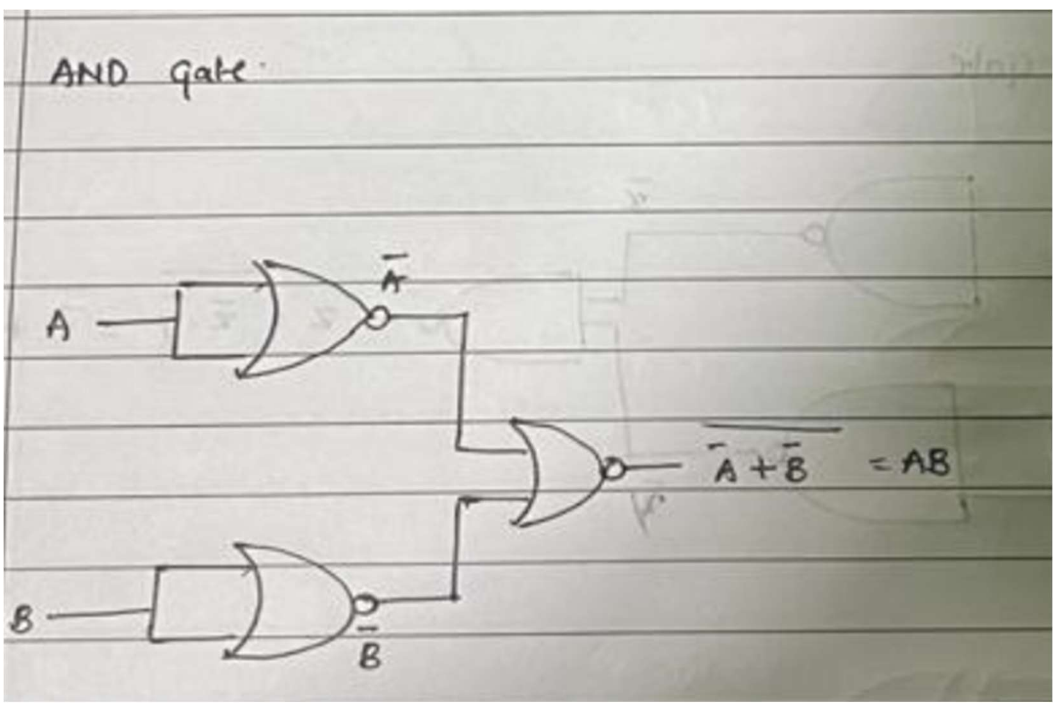
NOT GATE



AND GATE

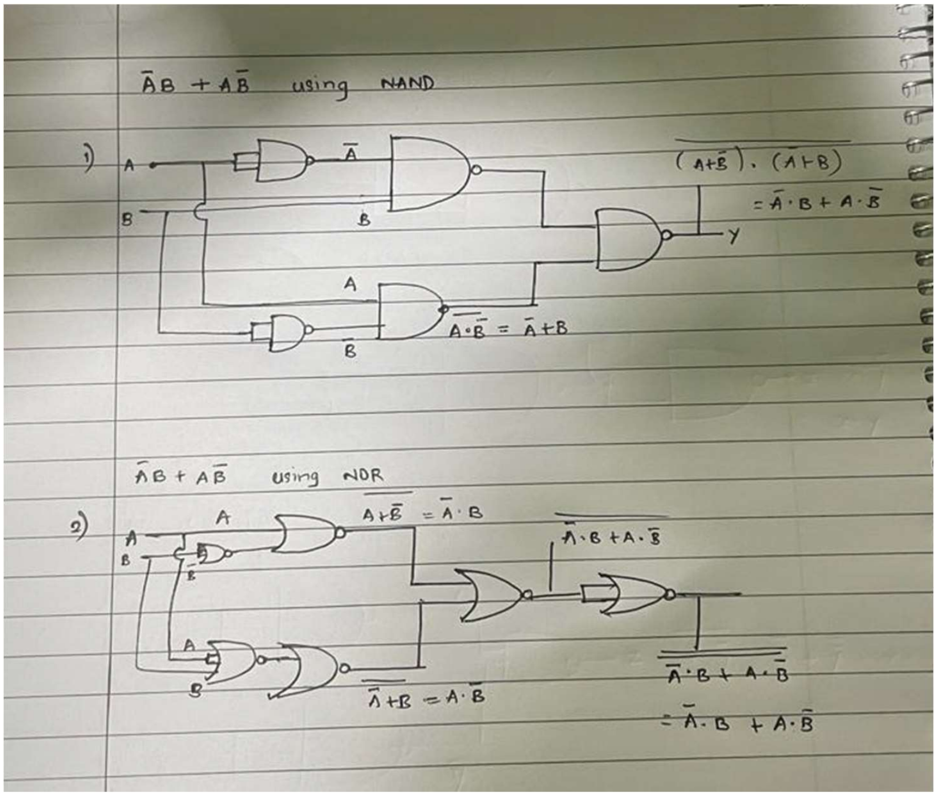


OR GATE

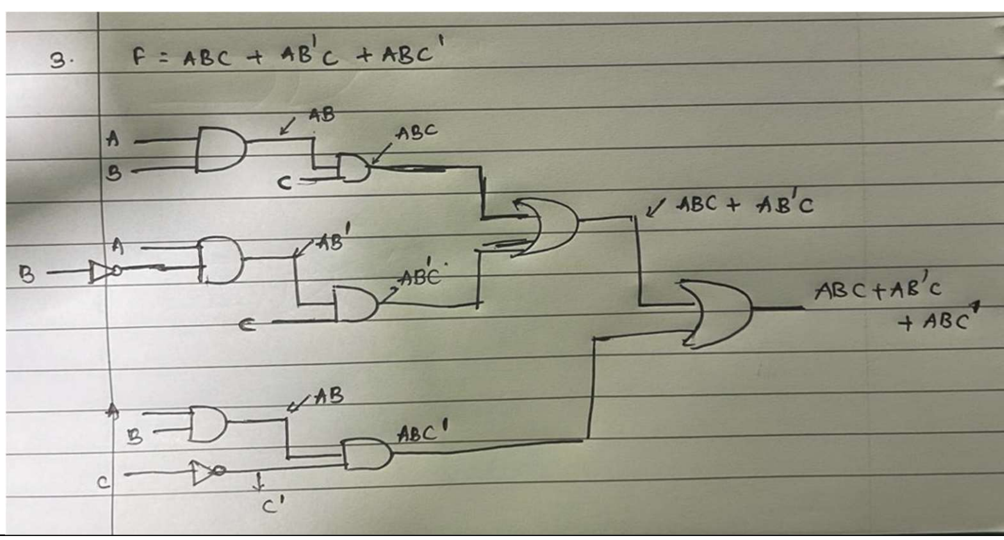


Post Lab Subjective/Objective type Questions:

1. Implement the Boolean function using NAND gates and NOR gates F=A’B + AB’



2. Implement using combination of gates F = ABC + AB’C + ABC’



Conclusion:
Learned about using Logic gates IC on the kit and also how to do implementations of equations using logic gates.

Signature of faculty in-charge with Date: