# 

# Submitted By:

# 

|  |  |  |
| --- | --- | --- |
| **Name** | **ID** | **Contribution** |
| NUSHRAT JAHAN | **22-46149-1** |  |
| S.M. MUJAHID SOUROV | **22-49679-3** |  |
| TRIDIB SARKAR | **22-46444-1** |  |
| MD. ATIK ISHRAK SUJON | **22-46684-1** |  |
| MD. FAHIM MURSHED | **22-46695-1** |  |

Date of Submission: , 2023

# TABLE OF CONTENTS

|  |  |
| --- | --- |
| **TOPICS** | **Page no.** |
| 1. **Cover Page** |  |
| **II. Table of Contents** |  |
| **Abstract** |  |
| **Theory** |  |
| **Apparatus** |  |
| **Experimental Procedure** |  |
| **Circuit Diagram** |  |
| **Images** |  |
| **IC Configuration** |  |
| **MULTISIM** |  |
| **Discussion** |  |
| **References** |  |
| **Questions & Answers** |  |

**Title:** Deriving logic equations and truth table from a given statement or expression and construction of combinational circuits

**ABSTRACT:**

This experiment is designed to-

1. Help students implement the logic circuits derived from a given statement in the breadboard

using gate ICs and observe whether the output verifies the truth table of the given logic statement

or not.

2. Perform relevant theoretical work by deriving the logic circuit and truth table from the given

logic equation/statement and get familiarized with Boolean algebra and De Morgan’s law.

3. Simplify the logic expressions with K-Map and verify accuracy by breadboard

implementation.

**INTRODUCTION:**

From any given logic statement, it is possible to construct a digital logic circuit. The first step in this process is to construct a truth table and then determine a standard SOP (sum of products) or POS (product of sums). At the same time, it is also possible to derive a logic expression from a given combinational circuit diagram by observing the individual logic operations performed in the circuit and matching them with their corresponding logic gates. Expressions are simplified using Boolean algebra and De Morgan's law or K-Map to reduce the number of gates used. Then the circuit is implemented in the breadboard using gate ICs and observed whether the output verifies the truth table of the given statement.

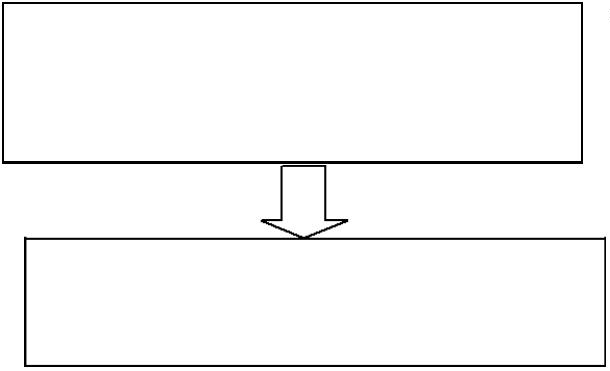
This experiment shows the students a practical verification of deriving logic equations and truth table from combinational circuits. Knowing how to derive logic equations and truth table from combinational circuits helps a person with detecting the output logic expressions from any unknown logic circuit.

# THEORY & METHEDOLOGY:

Combinational circuits are built with logic gates and other components. It does not include any

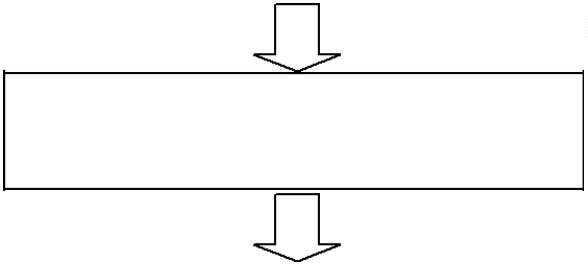
values to be taken from a previous state of the circuit. Designing such a combinational digital

system requires use of one of the following methods:

1. If a problem statement is given, the following steps will help designing the system

Look for the problem statement

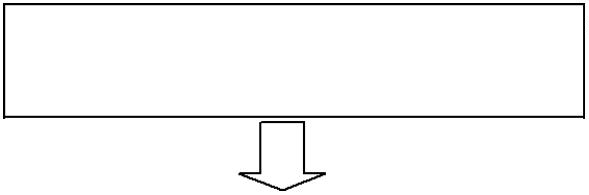
Find the input(s) and output(s) of the system and relate the input(s) with the output(s)



Develop a truth table for your system

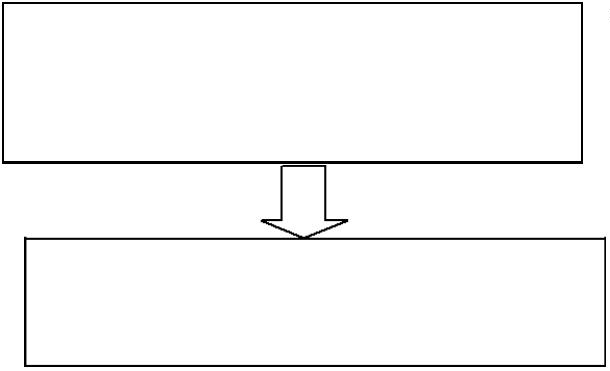
using the input and output relationship

that you have established



From your truth table, generate a

standard output expression



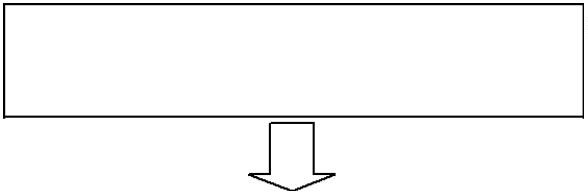
Reduce/simplify the output expression

using Boolean algebra or K-Map

Implement the circuit using the

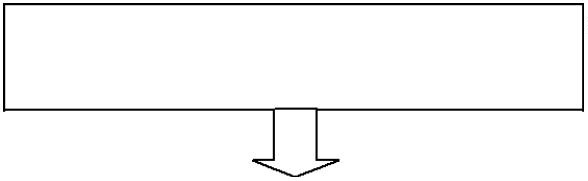
simplified Boolean expression.

1. Or if an expression is given, the following steps will help in designing the system



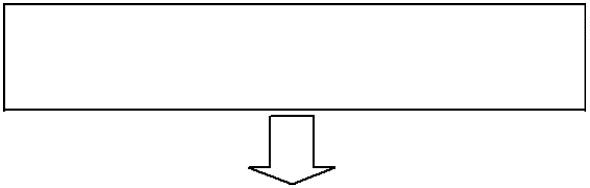
If the provided expression

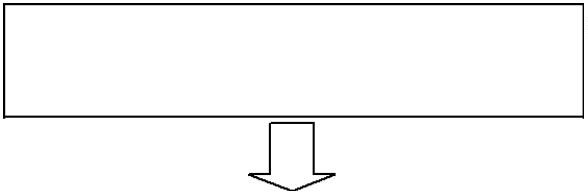
is not standard, make it standard



After standardization, create

a truth-table.

Use the truth-table to develop a K-MAP



Use proper grouping in K-MAP in

determining the output expression.

Use the reduced expression for

combinational circuit design. 

Some useful definitions related to these procedures are given below:

**Boolean algebra:** Boolean Algebra is a mathematical system based on logic that utilizes a set of rules and laws to simplify and reduce complex Boolean expressions, enabling the analysis and optimization of digital gates and circuits with variables that can only take on the values of 0 or 1. [1]

**1. Variable:** Boolean variables, such as A, B, and C, referred to as literals, can be represented by any symbol and can only have values of 0 or 1. [2]

**2. Complement:** The complement is defined as the inverse of a variable, which is represented by a bar over the variable. [1]

**3. Sum term:** The OR function, representing addition in Boolean Algebra, is called a sum term and is symbolized by the plus sign (+). [1]

**4. Product term:** the "product" function refers to the AND operation of terms with variables or constants, resulting in minterms that cannot be further simplified, and various combinations of these can generate product results. [1]

**5. Sum of Products (SOP):**

When two or more product terms are summed by boolean addition, the resulting expression is asum of product. Implementing an SOP expression simply requires ORing the outputs of two or more AND gates. Product term is produced by an AND operation, and the sum (addition) of two or more product terms is produced by an OR operation. Therefore, an SOP expression can be implemented by AND-OR logic in which the outputs of a number (equal to the number of product terms in theexpression) of AND gates connect to the inputs of an OR gate. A standard SOP expression is one in which all the variables in the domain appear in each product term. Ex. Standard SOP expressions are important in constructing truth-tables and in Karnaugh map simplification method. The SOP expression is equal to 1 only if one or more of the product terms in the expression is equal to 1. [1]

**6. Product of Sums (POS):**

the OR function produces the logical sum of Boolean addition, and that the AND function produces the logical sum of Boolean multiplication. But when dealing with combinational logic circuits in which AND gates, OR gates and NOT gates are connected together, the expressions of Product-of-Sum are widely used. The Product of Sum (POS) expression comes from the fact that two or more sums (OR’s) are added (AND’ed) together. That is the outputs from two or more OR gates are connected to the input of an AND gate so that they are effectively AND’ed together to create the final (OR AND) output. [1]

**7. Karnaugh Map:**

Karnaugh map or K-map is a map of a function used in a technique used for minimization or simplification of a Boolean expression. It results in less number of logic gates and inputs to be used during the fabrication.Booleans expression can be simplified using Boolean algebraic theorems but there are no specific rules to make the most simplified expression. However, K-map can easily minimize the terms of a Boolean function.Unlike an algebraic method, K-map is a pictorial method and it does not need any Boolean algebraic theorems. K-map is basically a diagram made up of squares. Each of these squares represents a min-term of the variables. If n = number of variables then the number of squares in its K-map will be 2n. K-map is made using the truth table. In fact, it is a special form of the truth table that is folded upon itself like a sphere. Every two adjacent squares of the k-map have a difference of 1-bit including the corners. Karnaugh map can produce Sum of product (SOP) or product of Sum (POS) expression considering which of the two (0,1) outputs are being grouped in it. The grouping of 0’s result in Product of Sum expression & the grouping of 1’s result in Sum of Product expression. The expression produced by K-map may be the most simplified expression but not unique. There can be more than 1 simplified expression for a single function but they all perform the same. [1]

**Problem1.** A Building has 4 floors which share the same water tank for water supply. In order to start the motor, each floor has a designated switch- Ground Floor with switch A, 1st Floor with switch B, 2nd Floor with switch C and 3rd Floor with switch D. The motor starts if someone

presses the switch from the 3rd floor or from both ground and 2nd floor or from 1st and 2nd floor. Your job is to design the system.

**Problem2.** For the expression and the logic gate diagram, find the truth-table, reduced expression

using K-MAP, and the logic gate diagram.

# Apparatus:

* Digital trainer board
* IC 7432:1 pcs
* IC 7408:1 pcs
* IC 7404:2 pcs
* IC 7402:1 pcs
* IC 7400:1 pcs
* IC 7486:1 pcs
* Connecting wires

# Experimental Procedure:

**Problem1:**

* We drew a truth table to represent the output Y.
* We used the truth table outputs to form standard SOP expressions.
* We minimized the SOP expression using Boolean algebra and K-Map. Perform hardware implementation of the circuit and compare with our truth table output.

**Problem2:**

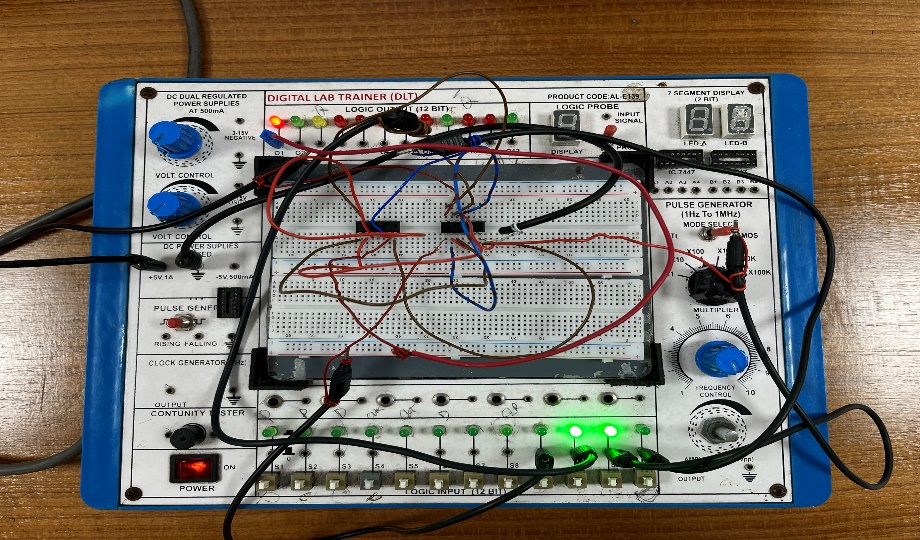
* We drew a step-by-step truth table to represent the outputs at each gate and then the final output at Y.
* We used the output Y to form standard SOP expression.
* We minimized the SOP expression using Boolean algebra and K-Map then we performed hardware implementation of the circuit and compared with the truth table output.

**Simulation and Measurement:**

# Experimental Circuits:

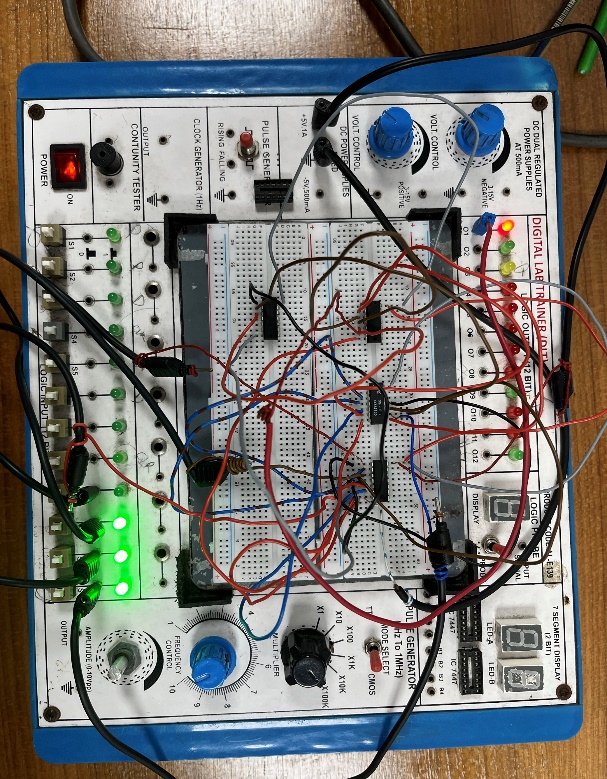
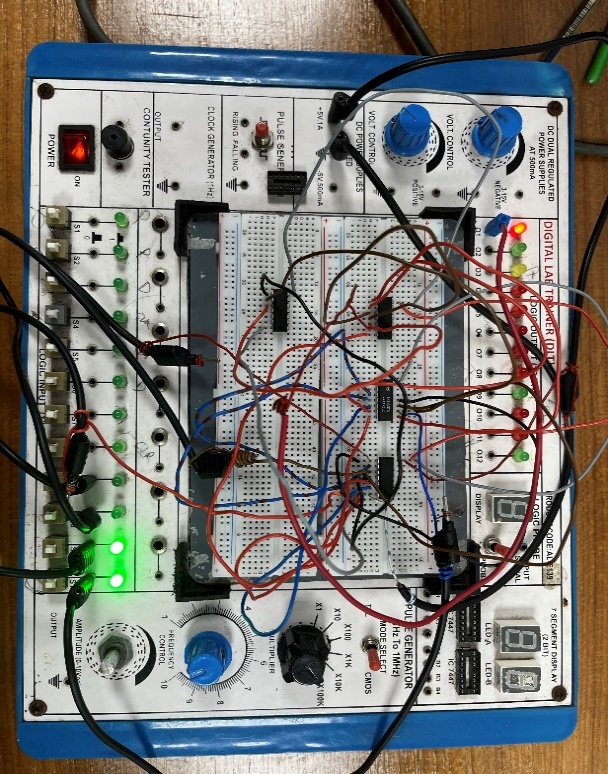
## **Problem 1:**

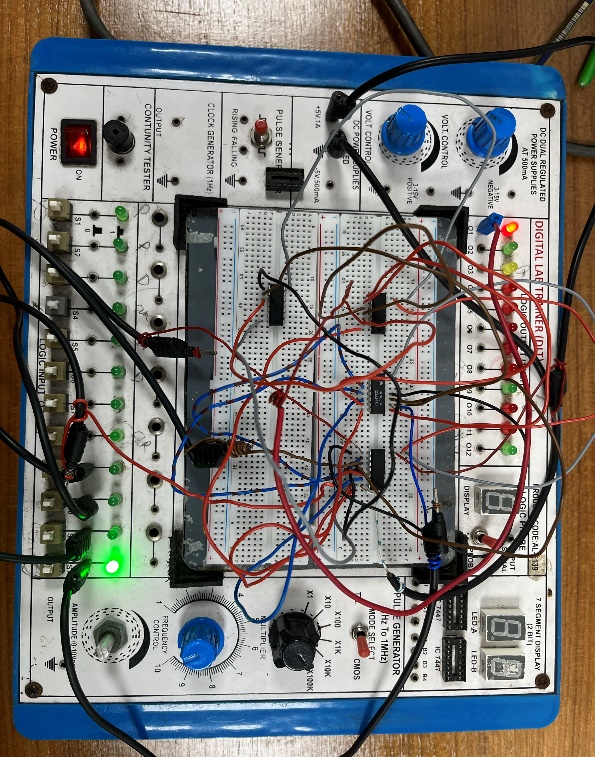
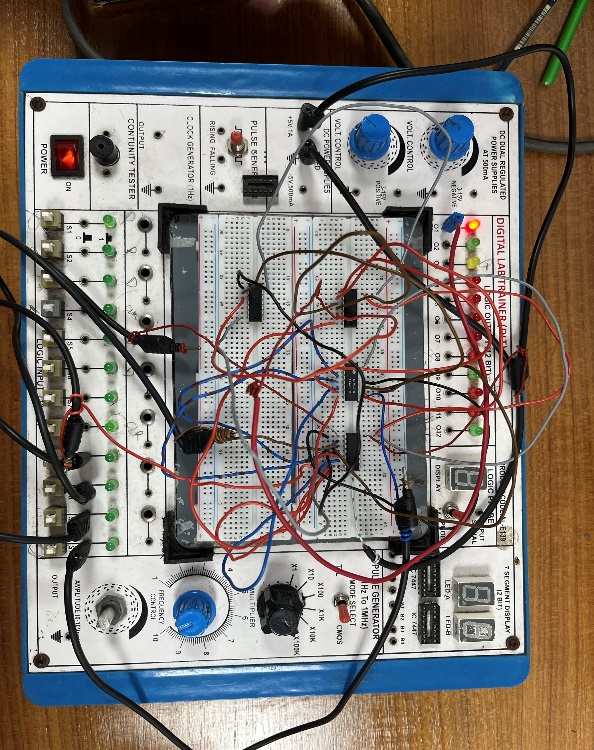
## 

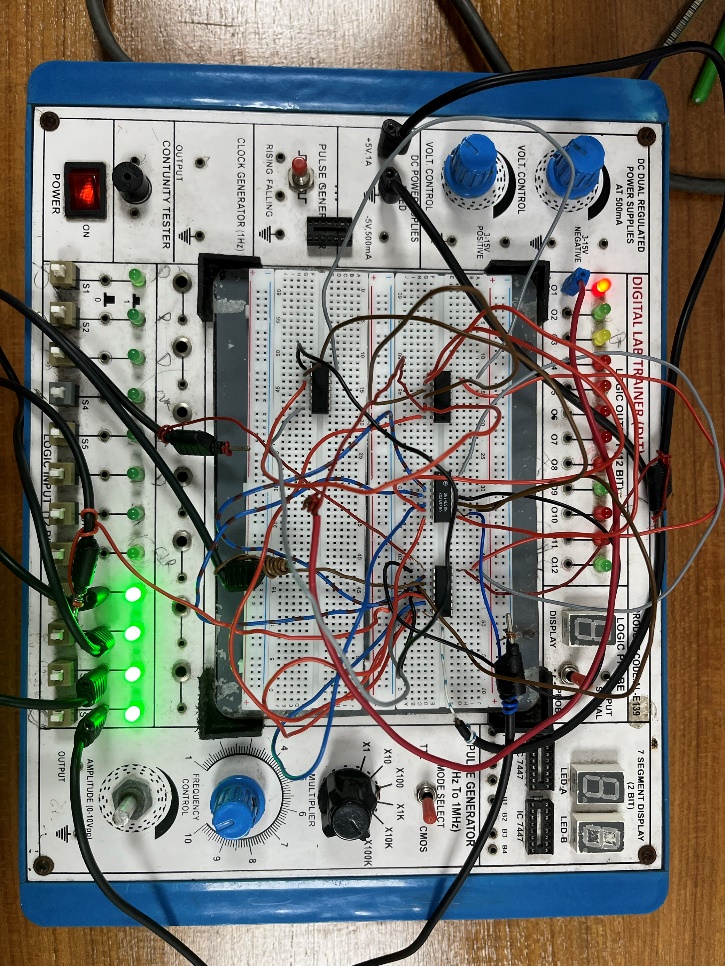


## **Problem 2:**

## 







# Results & Discussion:

In this lab experiment, we began by analyzing a specific problem and proceeded to generate a truth table that mapped the relationship between inputs and outputs. Using the truth table as a guide, we utilized Karnaugh Maps to simplify the Boolean expression, ultimately obtaining a Sum of Products (SOP) expression. With the SOP expression in hand, we constructed a circuit using appropriate components. This part of the experiment was successfully executed, resulting in the desired output. To further verify our work, we employed Multisim software. Initially, we encountered some errors during the verification process, but with diligence, we were able to identify and rectify them, ultimately achieving accurate results. Overall, this experiment demonstrated the effectiveness of our methodology in solving the problem at hand and highlighted the importance of careful analysis, circuit construction, and software validation in achieving reliable results.

# Reference:

# Thomas L. Floyd, “Digital Fundamentals”, available Edition, Prentice Hall International Inc.

**Report Questions:**

1. Construct the derived equations (i) and (ii), using Universal gates (both NAND and NOR).
2. Develop the truth table for a certain three-input logic circuit with the output expression
   1. Y=ABC+(AB)’C+A’BC+AB’C+A(B’+C).
3. Implement the following logic expressions with logic gates Y=ABC+AB+AC

**Report Answers:**

**Ans 1: Implementation of Y = D + C (A + B) using NAND and NOR gate only -**

**Figure 2:** Implementation using NOR

**Figure 1:** Implementation using NAND

**Ans 2:** After simplifying the expression:

Y = ABC+(AB)’C+A’BC+AB’C+A(B’+C)

= ABC + (A’+B’) C+ A’BC+ AB’C+AB’+AC

= ABC + A’C + B’C + A’BC+ AB’C+AB’+AC

= ABC+ A’BC + A’C+ AC+B’C+ AB’C+ AB’

= BC (A+A’) + C (A’+A) + B’C(1+A) + AB’

= BC.1+ C.1+B’C.1+ AB’ = BC+C+B’C+AB’

= C(B+1) + B’C+AB’ = C.1+ B’C+AB’

= C(1+B’) + AB’

= C+ AB’

Truth Table for the expression:

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **C** | **Y** |
| **0** | **0** | **0** | **0** |
| **0** | **0** | **1** | **1** |
| **0** | **1** | **0** | **0** |
| **0** | **1** | **1** | **1** |
| **1** | **0** | **0** | **1** |
| **1** | **0** | **1** | **1** |
| **1** | **1** | **0** | **0** |
| **1** | **1** | **1** | **1** |

**Ans 3:**

**Figure 1:** For Input is A = 1, B= 1, C = 0; Output is 1