|  |
| --- |
| **Digital Logic Design**  **(EL-1005)** |
| **LABORATORY MANUAL**  **Spring-2025** |

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LAB 04**  **Simplification of Digital Circuits Using Karnaugh map** | | | | |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | \_\_\_\_\_\_\_\_\_\_ | | \_\_\_ |
| STUDENT NAME | | ROLL NO | | SEC |
|  | | | | |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | |
| INSTRUCTOR SIGNATURE & DATE | | | | |
| **MARKS AWARDED: /10** | | | | |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | |
| **NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES (NUCES), KARACHI** | | | | |
|  | | | | |
|  |  |  |  | |

**Lab Session 04: Simplification of Digital Circuits Using Karnaugh Map**

**OBJECTIVES:**

The objectives of this lab is:

* To learn K-map and its usage in order to obtain cost effective circuit for implementation
* Implementation of Digital Circuits on Logisim/LogicWorks.

**SOFTWARE:**

* Logisim/LogicWorks

**Introduction:**

De-Morgan ‘s laws provide mathematical verification of the equivalency of the NAND and negative‐OR gates and the equivalency of the NOR and negative‐AND gates. The complement of a product of variables is equal to the sum of the complements of the variables. The complement of two or more AND variables is equivalent to the OR of the complements of the individual variables. The De Morgan’s statements are,

#### **Statement 1:**

“The negation of conjunction is the disjunction of the negations”. Or we can define that as “The compliment of the product of 2 variables is equal to the sum of the compliments of individual variables”.

(A.B)’ = A’ + B’

#### **Statement 2:**

“The negation of disjunction is the conjunction of the negations”. Or we can define that as “The compliment of the sum of two variables is equal to the product of the compliment of each variable”.

(A + B)’ = A’. B’

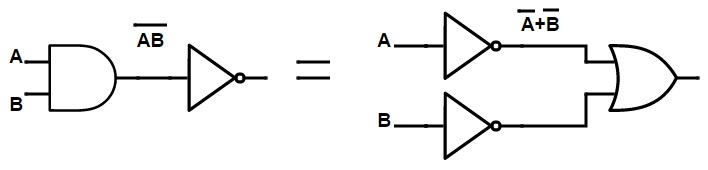
[](https://www.electronicshub.org/wp-content/uploads/2015/08/NAND-gate-equivalent-to-an-inversion-followed-by-OR-Gate.jpg)Figures of the about two statement shown below:

Figure 1 : NAND gate= Bubbled OR gate

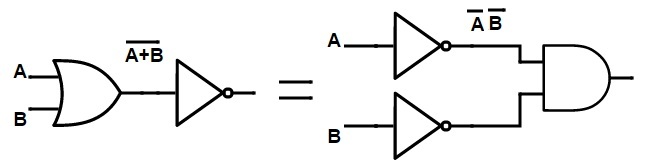
[](https://www.electronicshub.org/wp-content/uploads/2015/08/NOR-gate-equivalent-to-an-inversion-followed-by-AND-gate1.jpg)

Figure 2 : NOR gate= Bubbled AND gate

**Simpler expressions yield simpler hardware:**

The proof is shown in table, which shows the truth table and the resulting logic circuit simplification.

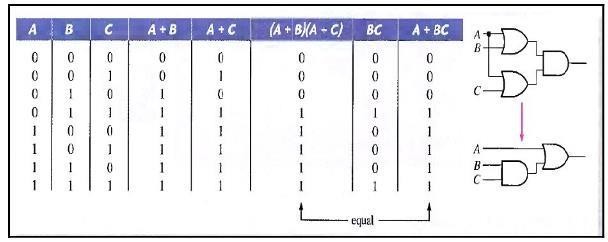


Figure 2 : Simplification of circuit

**Universality of logic Gates:**

**1. The NAND Gate as a Universal Logic Element**

Any logic expression can be implemented using only NAND gates or only NOR gates and no other type of gate. NAND gates alone in the proper combination, can be used to perform each of the basic Boolean operations OR, AND, and INVERT.

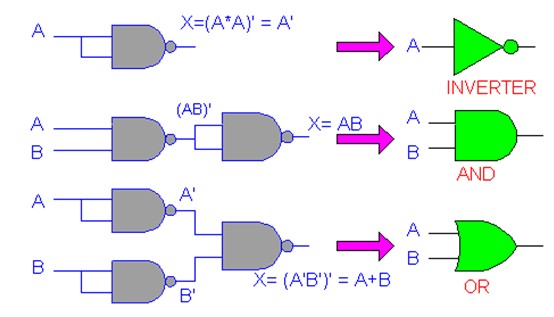


Figure 3 : NAND gate based Basic Gates

1. **The NOR Gate as a Universal Logic Element**

It can be shown that NOR gate can be arranged to implement any of the Boolean operations.

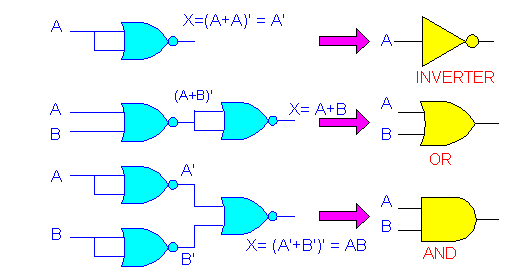
****

Figure 4 : NOR gate based Basic Gates

**SOP AND POS Boolean Expressions:**

1. **The Sum-of-Products (SOP) Form:**

When two or more product terms are summed by Boolean addition, the resulting expression is a sum-of-products (SOP).

Examples 01:

1. AB + ABC
2. ABC + CDE + BCD
3. AB + ABC + AC
4. **The Product-of-Sums (POS) Form:**

When two or more sum terms are multiplied, the resulting expression is a product-of-sums (POS).

Examples 02:

1. (A + B)(A + B + C)
2. (A + B + C)(C + D + E)(B + C + D)
3. (A + B)(A + B + C)(A + C)

**Domain of a Boolean Expression:**

The domain of a general Boolean expression is the set of variables contained in the expression in either complemented or uncomplemented form. For example, the domain of the expression AB + ABC is the set of variables A, B, C and the domain of the expression ABC + CDE + BCD is the set of variables A, B, C, D, E.

**Standard Forms of Boolean Expressions:**

1. **The Standard SOP Form:**

A standard SOP expression is one in which all the variables in the domain appear in each product term in the expression. We can convert any Non-standard SOP expression to a standard SOP by following the steps below:

Step 1: Multiply each nonstandard product term by a term made up of the sum of a

missing variable and its complement. This results in two product terms. As you

know, you can multiply anything by 1 without changing its value.

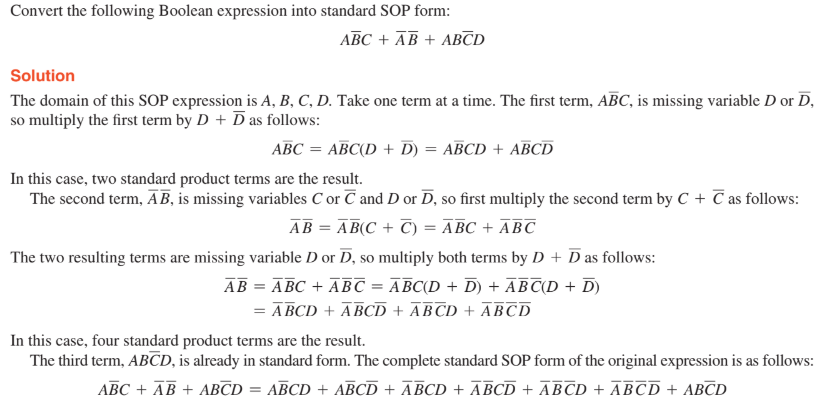
Step 2: Repeat Step 1 until all resulting product terms contain all variables in the

domain in either complemented or uncomplemented form. In converting a

product term to standard form, the number of product terms is doubled for each

missing variable.

**Example 03:**



1. **The Standard POS Form**

A standard POS expression is one in which all the variables in the domain appear in each sum term in the expression. We can convert any Non-standard POS expression to a standard POS by following the steps below:

Step 1: Add to each nonstandard product term a term made up of the product of the

missing variable and its complement. This results in two sum terms. As you

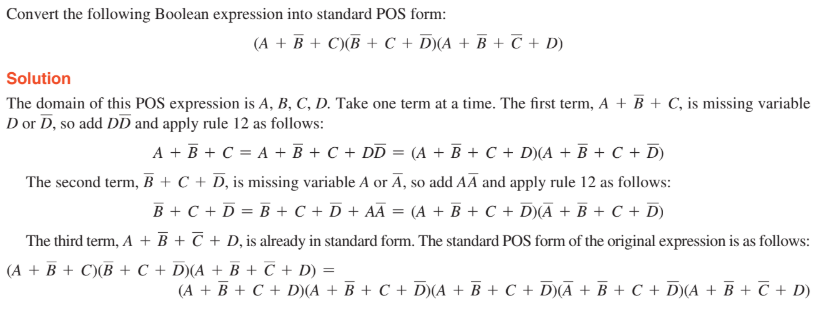
know, you can add 0 to anything without changing its value.

Step 2: Apply rule: A + BC = (A + B)(A + C)

Step 3: Repeat Step 1 until all resulting sum terms contain all variables in the domain

in either complemented or uncomplemented form.

**Example 04:**

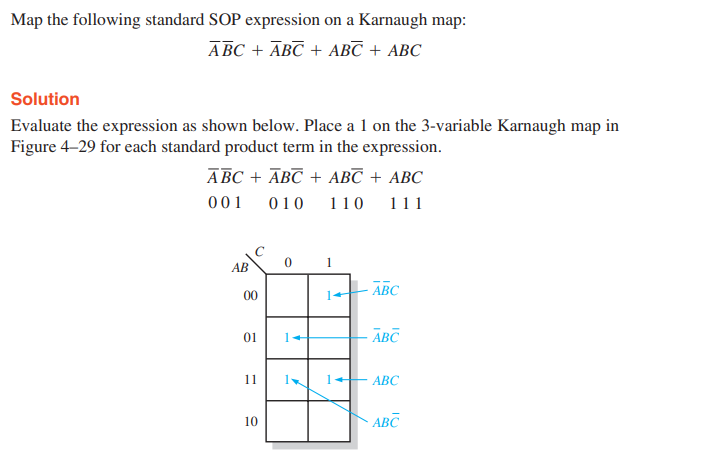


**K-MAP:**

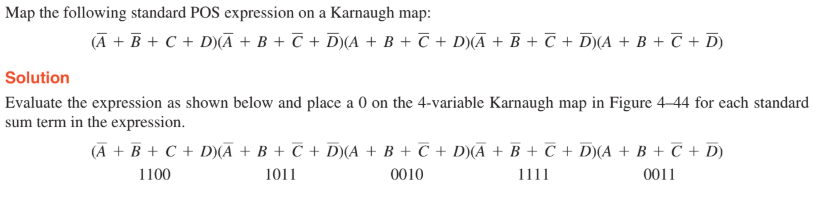
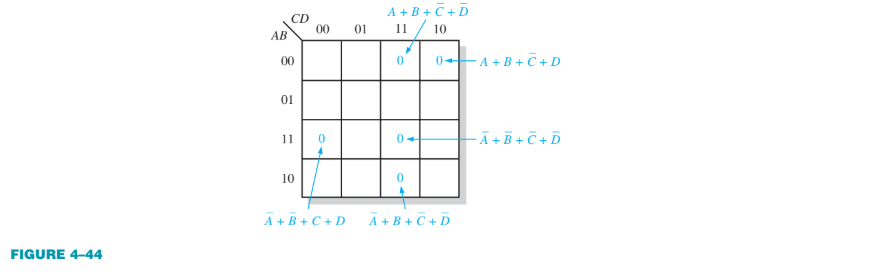
A Karnaugh map provides a systematic method for simplifying Boolean expressions and, if properly used, will produce the simplest SOP or POS expression possible, known as the minimum expression. As you have seen, the effectiveness of algebraic simplification depends on your familiarity with all the laws, rules, and theorems of Boolean algebra and on your ability to apply them. The Karnaugh map, on the other hand, provides a “cookbook” method for simplification. Karnaugh maps can be used for expressions with two, three, four, and five variables, but we will discuss only 3-variable and 4-variable situations to illustrate the principles. The number of cells in a Karnaugh map, as well as the number of rows in a truth table, is equal to the total number of possible input variable combinations. For three variables, the number of cells is 2^3 = 8. For four variables, the number of cells is 2^4 = 16.

Karnaugh map is used to obtain optimized logic representation so that it can be implemented using a minimum number of logic gates. The sum-of-product form can always be implemented using AND gates feeding into an OR gate, and a product-of-sum form leads to OR gates feeding an AND gate.

**SOP Expression Mapping on K-Map (3-Variables):**



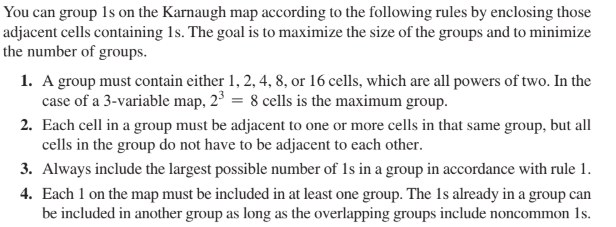
**POS Expression Mapping on K-Map (4-Variables):**

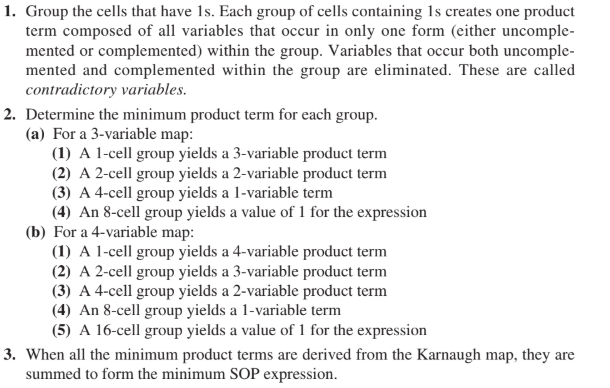
**Karnaugh Map Simplification of SOP Expressions:**

A minimum SOP expression is obtained by **grouping the 1s** and **determining the minimum SOP expression** from the map.

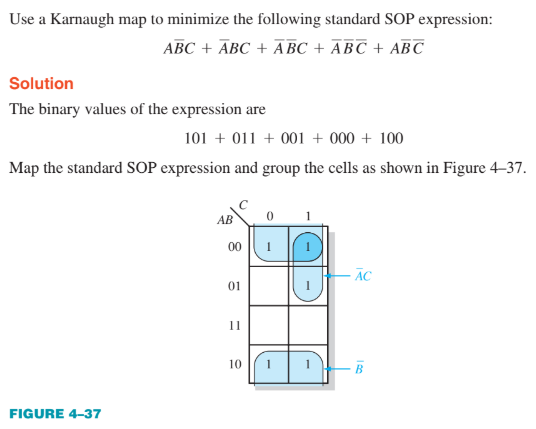
1. **Grouping the 1s**



1. **Determining the Minimum SOP Expression from the Map**



**Example 05:**

**Karnaugh Map Simplification of POS Expressions:**

The process for minimizing a POS expression is basically the same as for an SOP expression except that you group 0s to produce minimum sum terms instead of grouping 1s to produce minimum product terms. The rules for grouping the 0s are the same as those for grouping the 1s

**Example 06:**

