Digital Logic Design
(EL-227)
LABORATORY MANUAL
Spring-2021

LAB 04
Simplification of Digital Circuits

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES (NUCES), KARACHI

Date: 9th March 2021

Lab Session 04: Simplification of Digital Circuits

OBJECTIVES:

The objectives of this lab is:

- To learn K-map and its usage in order to obtain cost effective circuit for implementation
- Simplification of circuits using De-Morgan's Theorem

APPARATUS:

- Logic trainer
- Logic probe

COMPONENTS:

ICs 74LS02, 74LS00, 74LS08, 74LS32, 74LS04, Jumper Wire

BOOLEAN ALGEBRA

Boolean algebra is a set of rules used with digital variables to develop, manipulate and simplify logic expressions.

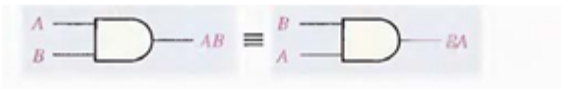
Rules and Laws of Boolean algebra

1. Commutative Laws:

The commutative law of addition for two variables is written as A+B=B+A

Application of commutative law of addition

The commutative law of multiplication for two variables is written as
AB=BA



Application of commutative law of multiplication

2. Associative Laws:

The associative law of addition is written as follows for three variables:

A + (B + C) = (A + B) + C

This law states that when ORing more than two variables, the result is the same regardless of the grouping of the variables.

Application of associative law of addition

The associative law of multiplication is written as follows for three variables:

A(BC) = (AB)C

This law states that it makes no difference in what order the variables are grouped when ANDing more than two variables.

Application of associative law of multiplication

3. Distributive Law:

The distributive law is written for three variables as follows:

$$A.(B + C) = AB + AC$$

This law states that ORing two or more variables and then ANDing the result with a single variable is equivalent to ANDing the single variable with each of the two or more variables and then ORing the products. The distributive law also expresses the process of factoring in which the common variable A is factored out of the product terms, for example,

4. Idempotent Property

$A+A = A$

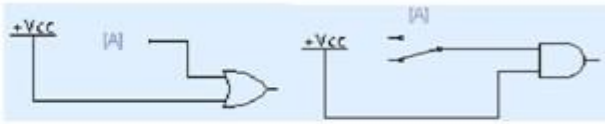
$A.A=A$

5. Negation Property

$$A + \overline{A} = 1$$
$$A.\overline{A} = 0$$

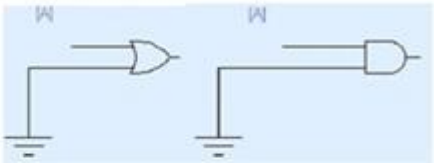
6. Identity Property

$$A + 1 = 1$$
$$A.1 = A$$



7. Null Property

$$A + 0 = A$$
$$A.0 = 0$$



8. Double Negation

$$\overline{\overline{A}} = A$$

Introduction to De-Morgan‘s laws:

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”.

$$(A + B)' = A'.B'$$

Truth Tables

The De Morgan’s laws are simply explained by using the truth tables. The truth table for De Morgan’s first Statement ((A.B)’ = A’ + B’) is given below.

A	B	A'	B'	A.B	(A.B)'	A'+B'
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

Table 1: Statement 1

The truth table for De Morgan’s second statement ((A + B)’ = A’.B’) is given below.

A	B	A'	B'	A+B	(A+B)'	A'.B'
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

Table 1: Statement 1

Figures of the about two statement shown below:

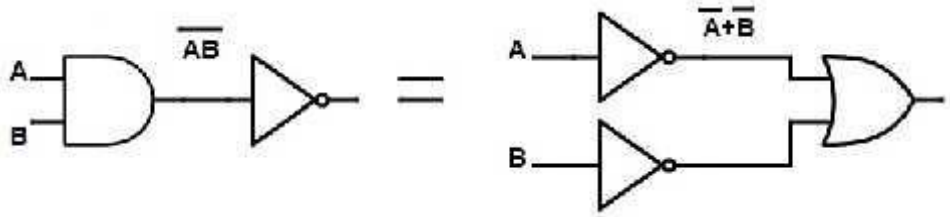


Figure 1 : NAND gate= Bubbled OR gate

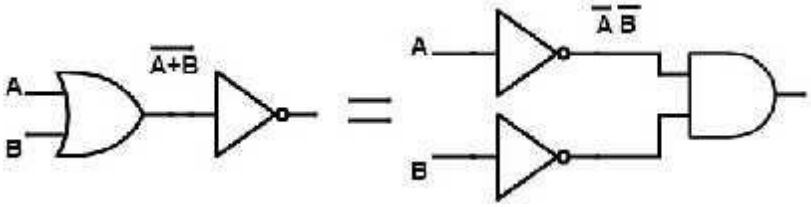


Figure 2 : NOR gate= Bubbled AND gate

Simpler expressions yield simpler hardware:

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

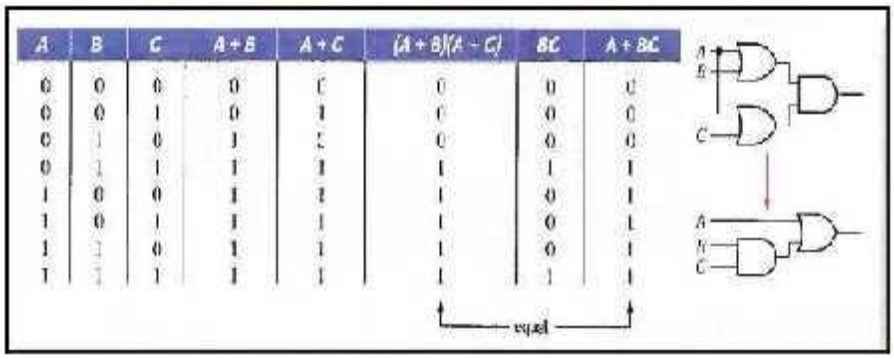


Figure 2 : Simplification of circuit

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.

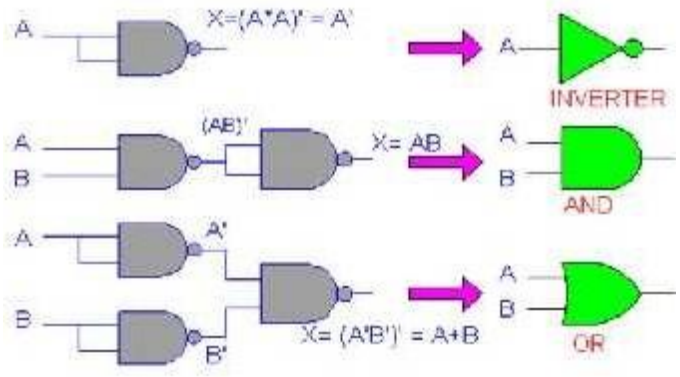


Figure 3 : NAND gate based Basic Gates

2. 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.

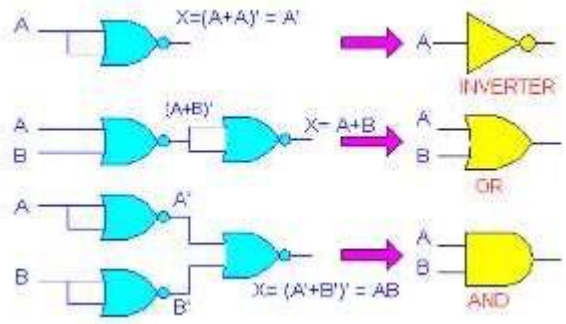


Figure 4 : NOR gate based Basic Gates

Lab Tasks# 04

Exercise # 01: Implement the following scenario/ Logic on Logic Works.

Two tanks store certain liquid chemicals that are required in a manufacturing process. Each tank has a sensor that detects when the chemical level drops to 25% of full. The sensors produce a HIGH level of 5 V when the tanks are more than one-quarter full. When the volume of chemical in a tank drops to one-quarter full, the sensor puts out a LOW level of 0 V.

It is required that a single red light-emitting diode (LED) on an indicator panel show when both tanks are more than one-quarter full. Show how a NAND gate can be used to implement this function.

Exercise#02: Analysis and Design Logic Circuit on Logic Works for the following scenario/ Logic.

For the process described in Exercise 01 it has been decided to have a red LED display come on when at least one of the tanks falls to the quarter-full level rather than have the green LED display indicate when both are above one quarter. Design circuit on logic works that shows how this requirement can be implemented.

Exercise#03: Analysis and Design Logic Circuit on Logic Works for the following scenario/ Logic.

As part of an aircraft's functional monitoring system, a circuit is required to indicate the status of the landing gears prior to landing. A green LED display turns on if all three gears are properly extended when the "gear down" switch has been activated in preparation for landing.

A red LED display turns on if any of the gears fail to extend properly prior to landing. When a landing gear is extended, its sensor produces a LOW voltage. When a landing gear is retracted, its sensor produces a HIGH voltage. Implement a circuit to meet this requirement.

Exercise#04:

Design circuit diagram in Logic Works given expressions by using either NAND or NOR gate.

1. **ABC + D' + E'**
2. **ABC + DE**

Exercise# 05:

Simplify the following Boolean expression and design circuit in Logic Works.

1. $\bar{A}BC + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} + A\bar{B}C + ABC$
2. $\overline{AB + AC} + \bar{A}\bar{B}C$

Basic rules of Boolean algebra.

1. $A + 0 = A$	**7.** $A \cdot A = A$
2. $A + 1 = 1$	**8.** $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	**9.** $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	**10.** $A + AB = A$
5. $A + A = A$	**11.** $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	**12.** $(A + B)(A + C) = A + BC$

Exercise# 06:

Apply De Morgan's theorems to each of the following expression and design circuit diagram on Logic Works.

(a) $\overline{(A + B + C)D}$

(b) $\overline{ABC + DEF}$

(c) $\overline{A\overline{B} + \overline{C}D + EF}$

Exercise# 07:

Convert the following Boolean expression into standard SOP form and design circuit diagram in Logic Works:

$$A\overline{B}C + \overline{A}\,\overline{B} + AB\overline{C}D$$

Exercise# 08:

Convert the following Boolean expression into standard POS form and design circuit diagram in Logic Works:

$$(A + \overline{B} + C)(\overline{B} + C + \overline{D})(A + \overline{B} + \overline{C} + D)$$