|  |
| --- |
| **Digital Logic Design**  **(EL-227)** |
| **LABORATORY MANUAL**  **Spring-2020** |

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LAB 06**  **Universal Logic Gates - Advance Logic Gate and Boolean Algebra** | | | | |
|  | |  | |  |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | \_\_\_\_\_\_\_\_\_\_ | | \_\_\_ |
| STUDENT NAME | | ROLL NO | | SEC |
|  | | | | |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | |
| INSTRUCTOR SIGNATURE & DATE | | | | |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | |
| **NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES (NUCES), KARACHI** | | | | |
| **Date: 17th Feb – 18th Feb 2020** | | | | |
|  |  |  |  | |

**Lab Session 05: Universal Logic Gates-Advance Logic Gate and Boolean algebra Simplification of Digital Circuits and Karnaugh Map**

**OBJECTIVES:**

The objectives of this lab is:

* To study the realization of basic gates using universal gates (NAND gate & NOR gate)
* To learn technology mapping **(NAND-NAND & NOR-NOR implementation)** and its significance in order to obtain cost effective circuit for implementation
* Investigate the logical properties of exclusive-OR, exclusive-NOR function and implement it using basic and universal gates.
* 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, ICs 74LS02, 74LS00, 74LS08, 74LS32, 74LS04, Jumper Wire, Logic Works

**Introduction:**

The design of a combinational circuit starts from the specification of the problem and culminates in a logic diagram or net-list that describes a logic diagram. The procedure involves the specification, formulation, optimization, & technology mapping.

Technology mapping is actually transformation of logic diagram or net-list to a new diagram using the available implementation technology. Typically NAND and NOR gates are more desirable to use in technology mapping due to the following reasons:

1. NAND and NOR gates are said to be universal gates where universal gate is a gate which can implement any Boolean function without needing any other type of gate.
2. Using universal gate in technology mapping may further reduce cost of optimized logic diagram.
3. Universal gates are easier to fabricate with electronic components.

A convenient way to implement a Boolean function with NAND gates only (NAND-NAND implementation) is to begin with the optimized logic diagram of the circuit consisting of AND, OR and NOT gates. The function is converted to pure NAND logic by replacing each gate in logic diagram with its representation using NAND gates only as shown in figure 5-1. After that, all inverter pairs are cancelled. The same conversion procedure is applied to implement a Boolean function with NOR gates only (NOR-NOR implementation).

**Universal Logic Gates:**

1. **NAND Gate:**

**“It is a device whose output is 1 if at least one or all of the inputs are low (0)”**

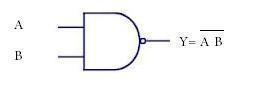
**Symbol:**

Figure 2 NAND Gate Symbol

**Function Table:**

|  |  |  |
| --- | --- | --- |
| **Inputs** | | **Output** |
| **A** | **B** | **Y** |
| L | L | H |
| L | H | H |
| H | L | H |
| H | H | L |

Table: 1 NAND Gate Truth Table

H= Logic High, L= Logic Low

#### **Connection Diagram:**

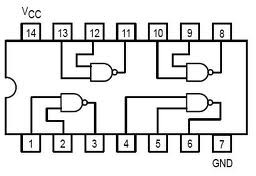
74LS00 IC contains four 2-input NAND gates. The connection diagram for this IC are shown below:

Figure 2 NAND Gate Connection diagram

1. **NOR Gate:**

***“It is a device whose output is 1 if all the given inputs are low (0)”.***

**Symbol:**

****

Figure 3 NOR Gate Symbol

**Function Table:**

|  |  |  |
| --- | --- | --- |
| **Inputs** | | **Output** |
| **A** | **B** | **Y** |
| L | L | H |
| L | H | L |
| H | L | L |
| H | H | L |

Table: 2 NOR Gate Truth Table

H= Logic High, L= Logic Low

#### **Connection Diagram:**

74LS02 IC contains four 2-input NOR gates. The function table and connection diagram for this IC are shown below:

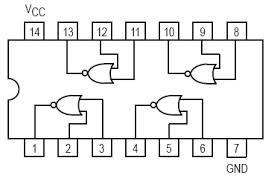


Figure 4 NOR Gate Connection diagram

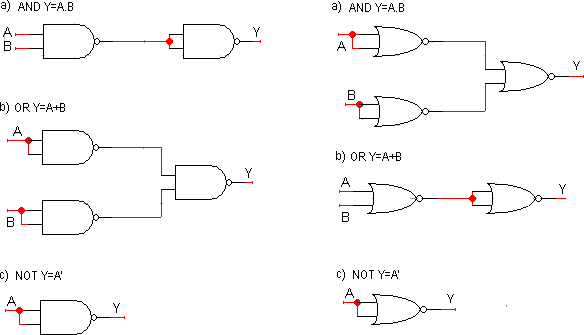
****

Figure 5 NAND-NAND and NOR-NOR representation of basic logic gates

**Application of Universal Gates:**

### Burglar alarm

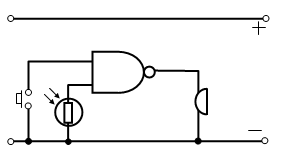
When the switch is closed one input of the NAND gate is LOW. When the LDR is in the light the other input is LOW. This means that if either of these things happen, i.e. the switch is closed or the light is on one of the inputs is LOW, the output is HIGH and the buzzer sounds.

Figure 6 NAND- Gate Based Burglar alarm

### Freezer warning buzzer

When the thermistor is COLD its resistance is LARGE and the input to the NAND gate is high.  
Since the NAND gate is connected as an INVERTER the output is LOW. As the thermostat warms up its resistance decreases, the voltage across it falls and the input to the NAND gate falls.

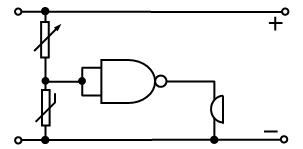
****

Figure 7 NAND- Gate Based Freezer warning buzzer

1. **Car Door Open System of an Automobile**

A car needs to be designed that the driver gets a visual indication if any of the doors of the car is open so that it helps to avoid accident and injury to the passengers. Assuming there are two doors (just for simplicity, it works for more doors as well) where this system is fitted, the circuit can be designed using a NAND gate as follows You can see from the figure that when any of the switches is open due to the door position, the NAND gate energies the lamp inside the car, hence warning the driver.

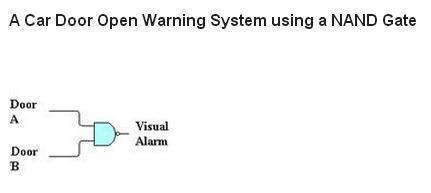
****

Figure 8 NAND- Gate Based Car Open System of an Automobile

Secondary gates can be made by the combinations of primary and universal gates. There are two types of secondary gates which may be termed as advanced gates,

1. **The Exclusive-OR Gate (XOR Gate):**

The exclusive OR function is an interesting and useful logical operation. As the name implies, it is similar to the previously studied OR function, but it’s a new and distinct operation. **"It is a device whose output is 1 only when the two inputs are different, but 0 if the inputs are the same."** This is useful for comparator circuits; if the inputs are different, then the output will be true, otherwise it is false.

**The symbol for exclusive-OR function is  and the logical expression is shown in fig below.**

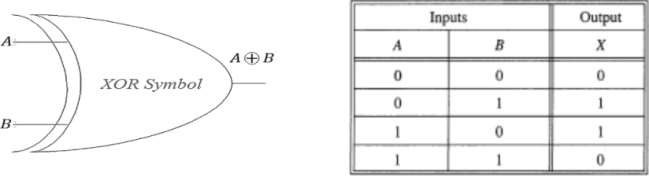
**Symbol:**

Figure 8 Exclusive-OR Gate Symbol

**Function Table:**

|  |  |  |
| --- | --- | --- |
| **Inputs** | | **Output** |
| **A** | **B** | **Y** |
| L | L |  |
| L | H |  |
| H | L |  |
| H | H |  |

Table: 1 XOR Gate Truth Table

H= Logic High, L= Logic Low

#### **Connection Diagram:**

7486 IC contains four 2-input XOR gates. The connection diagram for this IC is shown below:

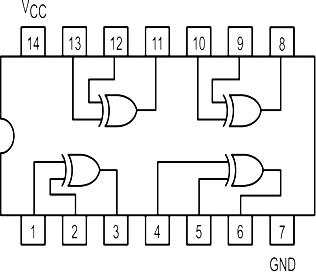
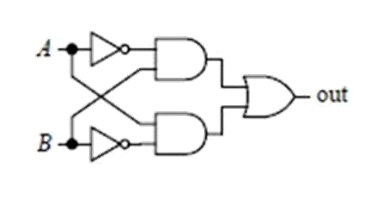
****

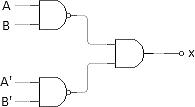
Figure 9 XOR Gate Connection diagram

The XOR gate can be implemented by using primary and universal gates as follows



A B = A'B+AB'

Figure 10 XOR Gate using basic Gate

****

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

Figure 11 XOR Gate using universal Gates

1. **Exclusive-NOR Gate (XNOR)**

An XNOR gate (sometimes referred to as Exclusive NOR gate) is a digital logic gate with two or more inputs and one output that performs logical equality. **The output of an XNOR gate is 1 when all of its inputs are same. If some of its inputs are 1 and others are 0, then the output of the XNOR gate is 0.**

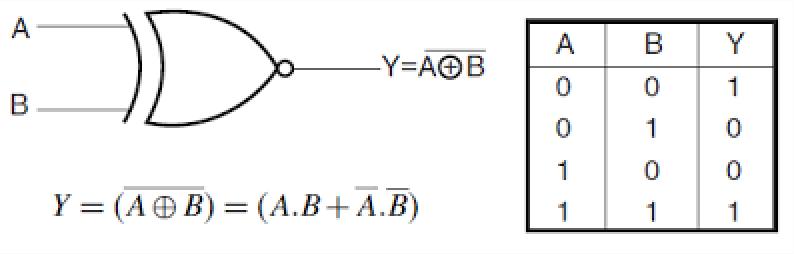
**Symbol:**

Figure 13 Exclusive-NOR Gate Symbol

**Function Table:**

|  |  |  |
| --- | --- | --- |
| **Inputs** | | **Output** |
| **A** | **B** | **Y** |
| L | L |  |
| L | H |  |
| H | L |  |
| H | H |  |

Table: 2 XNOR Gate Truth Table

H= Logic High, L= Logic Low

#### **Connection Diagram:**

74266 IC contains four 2-input XOR gates. The connection diagram for this IC is shown below:

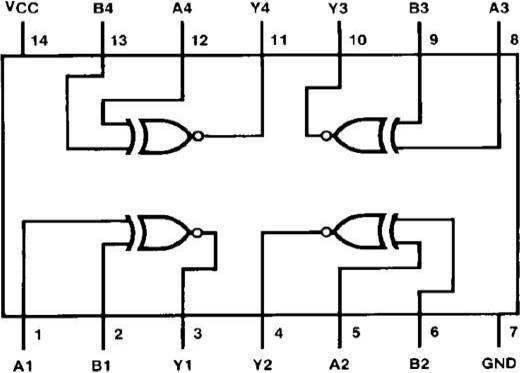


Figure 14 XNOR Gate Connection diagram

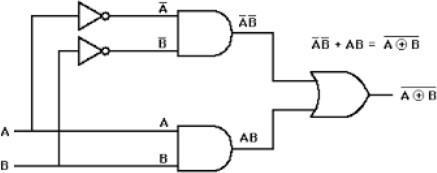
XNOR Gate can also be implemented by using primary gates as follows.

Figure 15 XNOR Gate implementation using primary gate

**Boolean Algebra:**

When a Boolean expression is implemented with logic gates, each term requires a gate, and each variable within the term designates an input to the gate. **Boolean algebra is applied to reduce an expression for obtaining a simpler circuit.** A Boolean function can be written in a variety of ways when expressed algebraically. There are, however, a few ways of writing algebraic expressions that are considered to be standard forms.

The standard forms contain product terms and sum terms. An example of a product term is XYZ. This is a logical product consisting of an AND operation among three literals. An example of a sum term is X+Y+Z. This is a logical sum consisting of OR operation among the literals.

**Rules and Law of Boolean Algebra:**

1. **Commutative law**

Commutative law states that the inter-changing of the order of operands in a Boolean equation does not change its result.

1. Using OR operator → A + B = B + A
2. Using AND operator → A \* B = B \* A

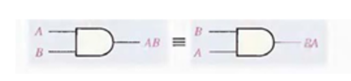


Figure 16 Commutative law in AND Gate

### Associative Law

### Associate Law of Addition:

### Associative law of addition states that OR more than two variables i.e. mathematical addition operation performed on variables will return the same value irrespective of the grouping of variables in an equation. It involves in swapping of variables in groups. The Associative law using OR operator can be written as

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

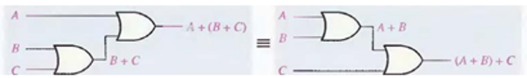


Figure 17 Application of Associative law of addition

#### **Associate Law of  Multiplication**

#### Associative law of multiplication states that AND more than two variables i.e. mathematical multiplication operation performed on variables will return the same value irrespective of the grouping of variables in an equation. The Associative law using AND operator can be written as

#### A \* (B \* C) = (A \* B) \* C

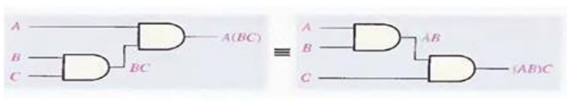


Figure 18 Application of Associative law of Multiplication

### Distributive law

### This is the most used and most important law in Boolean algebra, which involves in 2 operators: AND, OR. The multiplication of two variables and adding the result with a variable will result in same value as multiplication of addition of the variable with individual variables. Distributive law can be written as

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

This is called OR distributes over AND.

The addition of two variables and multiplying the result with a variable will result in same value as addition of multiplication of the variable with individual variables. Distributive law can be written as

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

This is called AND distributes over OR.

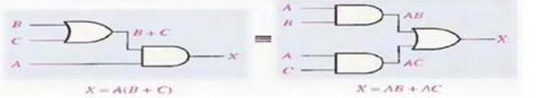


Figure 19 Application of Distributive law of Multiplication over addition and vice-versa

**De-Morgan‘s laws 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’

**Truth Tables**

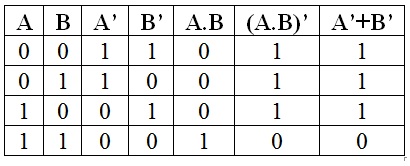
The De Morgan’s laws are simply explained by using the truth tables. The truth table for De Morgan’s first statement ((A.B)’ = A’ + B’) is given below.

Table 1: Statement 1

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

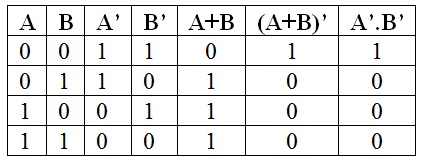
[](https://www.electronicshub.org/wp-content/uploads/2015/08/table-3.jpg)

Table 2: Statement 2

Figures of the about two statements shown below:

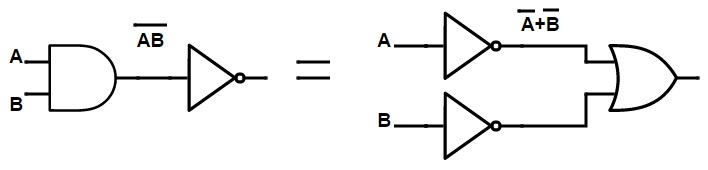
[](https://www.electronicshub.org/wp-content/uploads/2015/08/NAND-gate-equivalent-to-an-inversion-followed-by-OR-Gate.jpg)

Figure 1 : NAND gate= Bubbled OR gate

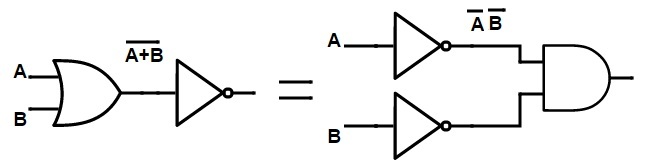
[](https://www.electronicshub.org/wp-content/uploads/2015/08/NOR-gate-equivalent-to-an-inversion-followed-by-AND-gate1.jpg)

Figure 20 : 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.

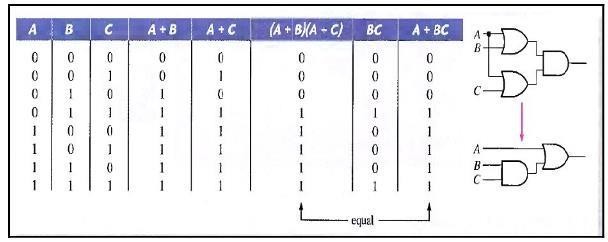


Figure 21 : Simplification of circuit

**K-MAP:**

The Karnaugh map (K-map) is a method used to simplify Boolean expressions. K-Map is a grid- like representation of a truth table that gives more insight. The required Boolean results are transferred from a truth table onto a two-dimensional grid where the cells are ordered in gray code and each cell position represents one combination of input conditions, while each cell value represents the corresponding output value. Optimal groups of 1s or 0s are identified, which represent the terms of a canonical form of the logic in the original truth table. These terms can be used to write a minimal Boolean expression representing the required logic.

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.

**Report for Experiment 05**

Name Student ID Section

**Exercise # 01**

Implement the logic gates (NAND-NOR-XOR- Gates) on logic works.

**Exercise # 02**

Implement the logic gate (XNOR- Gate) on logic works using primary logic gates.

**Exercise # 03**

Implement the basic logic gates (AND-OR-NOT Gates) using NAND gates only on logic works

**Exercise # 04**

Implement the logic gates (XOR-XNOR Gates) using NAND gates only on logic works

**Exercise # 05**

Implement the De-Morgan‘s laws both statements on logic works ***(A.B)’ = A’ + B' and (A+B)'=A'B'***

**Exercise # 06**

Implement the following expressions once by using NAND then NOR gate.

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

***2. ABC + DE***

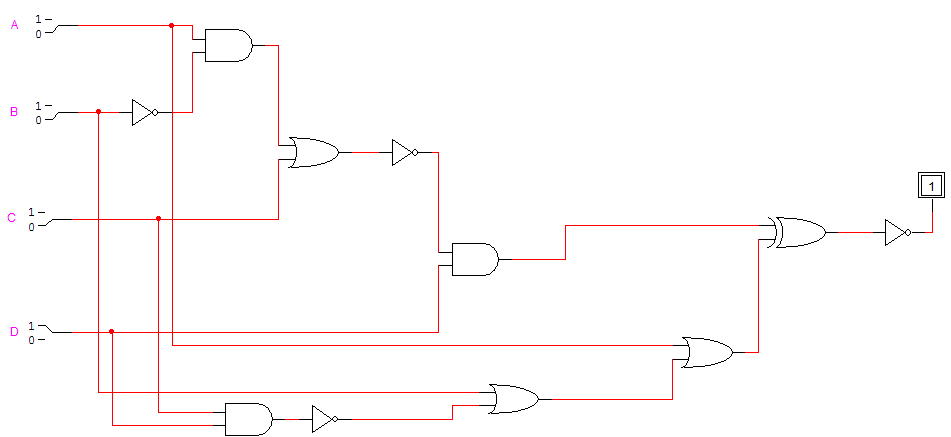
**Exercise # 07**

Implement on Logic works following circuits and write down its truth table.

1. ***A’BC + AB’C’ + A’B’C’ + AB’C + ABC***
2. ***XY'+Y'Z'+XYZ***
3. ***LM+M'(MN+N')***
4. ***ABC(A'+B'+C)***

**Exercise # 11**

**Write down its boolean expression also draw its complete truth table**

****

**QUOTE OF THE WEEK:**

**May you get all that you really deserve, be polite and observe, Give your best, and speak your mind, Success, You will definitely find, all the best!**