This Lab Manual is intended to support the students of undergraduate Engineering and Computer Science in the field of Digital Logic Design. This book begins with an introduction to the Logic Trainer and Electronic Workbench. Further, it covers experiments for basic logic gates, combinational circuits, functions of combinational circuits like adders, subtracters, decoders and multiplexers, sequential circuits, counters and registers. The purpose of experiments described here is to acquaint the students with the analysis and design aspects of combinational and synchronous sequential circuits.

Assistant Professor, Electrical Engineering Department, COMSATS Institute of Information Technology, Abbottabad, Pakistan.

978-3-330-05112-6

Digital Logic Design Lab Manual Volume I

Khan

Asadullah Khan

# Digital Logic Design, Lab Manual, Volume-I

LAP LAMBERT Academic Publishing

**Asadullah Khan**

# Digital Logic Design, Lab Manual, Volume-I

Asadullah Khan

# Digital Logic Design, Lab Manual, Volume-I

LAP LAMBERT Academic Publishing

**Impressum / Imprint**
Bibliografische Information der Deutschen Nationalbibliothek: Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.
Alle in diesem Buch genannten Marken und Produktnamen unterliegen warenzeichen-, marken- oder patentrechtlichem Schutz bzw. sind Warenzeichen oder eingetragene Warenzeichen der jeweiligen Inhaber. Die Wiedergabe von Marken, Produktnamen, Gebrauchsnamen, Handelsnamen, Warenbezeichnungen u.s.w. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutzgesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Bibliographic information published by the Deutsche Nationalbibliothek: The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.
Any brand names and product names mentioned in this book are subject to trademark, brand or patent protection and are trademarks or registered trademarks of their respective holders. The use of brand names, product names, common names, trade names, product descriptions etc. even without a particular marking in this work is in no way to be construed to mean that such names may be regarded as unrestricted in respect of trademark and brand protection legislation and could thus be used by anyone.

Coverbild / Cover image: www.ingimage.com

Verlag / Publisher:
LAP LAMBERT Academic Publishing
ist ein Imprint der / is a trademark of
OmniScriptum GmbH & Co. KG
Bahnhofstraße 28, 66111 Saarbrücken, Deutschland / Germany
Email: info@omniscriptum.com

Herstellung: siehe letzte Seite /
Printed at: see last page
**ISBN: 978-3-330-05112-6**

Copyright © Asadullah Khan
Copyright © 2017 OmniScriptum GmbH & Co. KG
Alle Rechte vorbehalten. / All rights reserved. Saarbrücken 2017

# TABLE OF CONTENTS


| TOPICS | | PAGE NUMBER |
|---|---|---|
| I. | PLAYING WITH ELECTRONIC WORKBENCH (EWB 5.12) AND LOGIC TRAINER | 03 |
| II. | BASIC LOGIC GATES (AND, OR, NOT) | 10 |
| III. | BOOLEAN ANALYSIS OF LOGIC CIRCUITS | 19 |
| IV. | BOOLEAN ALGEBRA AND SIMPLIFICATION OF BOOLEAN EXPRESSIONS | 26 |
| V. | UNIVERSAL GATES | 34 |
| VI. | ADDERS AND SUBTRACTORS | 44 |
| VII. | DECODERS | 53 |
| VIII. | MULTIPLEXERS | 60 |
| IX. | FLIP-FLOPS | 68 |
| X. | SYNCHRONOUS SEQUENTIAL CIRCUITS | 77 |
| XI. | COUNTERS | 86 |
| XII. | REGISTERS | 91 |
| XIII. | SEMESTER PROJECT | 95 |

# LAB-I

## PLAYING WITH ELECTRONIC WORKBENCH (EWB 5.12) AND DIGITAL LOGIC TRAINER

### DIGITAL LOGIC TRAINER:

**INTRODUCTION:**

Logic trainer is a low cost, high performance digital logic system. It is designed to provide all the basic tools necessary to conduct logic experiments.

**Specifications:**

- Input voltage 220V 50 Hz AC.
- Circuit type TTL compatible.
- 16 lamp monitors-transistor buffered.
- Logic probe with pulse detection.
- One clock timer.
- Output voltage +12V, -12V and +5V at IA.

**Components:**

- 16 LED's with drivers.
- 8 logic switches.
- 2 logic switches with debouncing circuitry.
- Logic probe with LOW, HIGH and PULSE indicators.
- 2 push pull switches to generate negative going pulse.
- One timer based on NE-555 with variable control.
- Three 7-segment displays with BCD decoders/drivers.
- One linear variable 10K.
- One linear variable 100K.
- Two breadboards.

**Power Supply:**

Power supply with short circuit protection, provides +5V, +12V and -12V with power indicators and fuses. Push-in wire terminals are provided for each supply.

**State Monitors:**

State monitors are light emitting diodes that are used to indicate the state of a logical output. Lighted diode represents a 1 and an unlighted diode represents a 0. They are labeled L0 through L15.

**Logic Switches:**

Logic switches S2 through S9 are special type of switches that are designed to produce two TTL compatible logic outputs, which are complements of each other. Switches S0 and S1 are simple logic switches, each having one output and are provided with debouncing circuitry.

**Digital Display:**

Three 7-segement decoders/drivers and displays are provided.

**Clock Timer:**

A 555-clock/timer circuit is used as a pulse source. The frequency can be adjusted by turning the variable 500K pot.

## ELECTRONIC WORKBENCH:

## INTRODUCTION:

Electronic Workbench is a simulation package for electronic circuits. It allows to design and analyze circuits without using breadboards, real components or actual instruments. EWB's click-and-drag operations make constructing and editing a circuit fast and easy. Circuit parameters and components can be changed on the fly, which makes the analysis straight forward and convenient.

GUI Interface of EWB 5.12

## BASIC FUNCTIONS:

### Selecting/Deselecting:

- The color of a selected component changes to red, indicating its selection.
- To select a single item, simply ***click*** on it.
- To select more than one items, press ***CTRL+click***.
- To select all, go to ***Edit>Select All*** or press ***CTRL+A***.
- Item(s) can also be selected by dragging the cursor.
- To deselect a single item, press ***CTRL+click***.
- To deselect all, click on any empty spot inside the window.

### Labeling:

To assign label to a particular component, select the component and go to ***Circuit>Component Properties..>Label***, or right click the component and

go to ***Component Properties..>Label***, or simply double click the desired component.

**Wiring:**

To wire components together, point to the terminal of the first component, when it gets highlighted, drag the cursor to meet the terminal of the second component.

To delete a wire, select it and go to ***Edit>Delete***, or select it and press ***DEL***, or select one end of it and move it to a vacant spot on the circuit window.

To change the color of a wire, double click it, a ***Wire Properties*** pop-up window will open, where you can choose your desired color from the ***Schematic Options*** tab.

**Inserting:**

To insert new component into existing circuit, place it on top of the wire, it will be automatically inserted, provided there is room for it.

**Simulation:**

Click the Power switch to turn power on, click it again to turn it off.

To pause the simulation click the Pause button.

## LAB TASKS:

### TASK-I: Name the Basic Toolboxes of EWB

## TASK-II: Familiarize With Basic Buttons In EWB Toolboxes

## TASK-III: Draw the following Circuit

- Connect the **Ground** to the input of the **Inverter**.
- Connect the output of the Inverter to the input of the **Red Probe.**
- Start **Simulation.**
- State your observations down.

## Observations:

Stop the simulation and connect **+Vcc Voltage Source** instead of Ground to the input of the Red Probe. Start simulation and state your observations.

**Observations:**

**TASK-IV: Name the following Icons and state down their functions**

_____

_____

_____

_____

# LAB-II

## BASIC LOGIC GATES (AND, OR, NOT)

### OBJECTIVES:

- To study and analyze the operations of three basic logic gates (AND, OR, NOT).
- Implement them in EWB and on Logic Trainer.
- Using logic converter, derive truth tables.

### INTRODUCTION:

**The NOT Gate:**

The NOT gate or Inverter performs the complementation operation. It converts 1 to 0 and 0 to 1. It has a single input and a single output. It's the only logic gate that has only one input. Logic symbol for NOT gate is shown below.

If it has A at its input, the output X will be its complement i.e. A'.

**The AND Gate:**

The AND gate performs the Boolean multiplication operation. It has more than one inputs and a single output. Logic symbol for a two input AND gate is shown below.

If the two inputs are A and B, the logic expression for output X will be:

**X = AB**

Output of an AND gate is HIGH, if and only if all its inputs are HIGH, else the output is LOW. Truth table for a two input AND gate is shown below.

| Inputs | | Outputs |
|---|---|---|
| **A** | **B** | **X** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**<u>The OR Gate:</u>**

The OR gate performs the Boolean addition operation. It has more than one inputs and a single output. Logic symbol for a two input OR gate is shown below.

If the two inputs are A and B, the logic expression for output X will be:

**X = A+B**

Output of an AND gate is LOW, if and only if all its inputs are LOW, else the output is HIGH. Truth table for a two input AND gate is shown below.

| Inputs | | Outputs |
|---|---|---|
| **A** | **B** | **X** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

## Pin Configuration of NOT, AND and OR ICs:

### 7404 (HEX INVERTER)

- **1A, 2A, 3A, 4A, 5A, 6A:** Inputs of INV-1, INV-2, INV-3, INV-4, INV-5 and INV-6 respectively.
- **1Y, 2Y, 3Y, 4Y, 5Y, 6Y:** Outputs of INV-1, INV-2, INV-3, INV-4, INV-5 and INV-6 respectively.
- **GND and Vcc:** Supply connection lines.

7404

### 7408 (Quad 2-In AND)

- **1A and 1B:** Inputs of AND-1.
- **2A and 2B:** Inputs of AND-2.
- **3A and 3B:** Inputs of AND-3.
- **4A and 4B:** Inputs of AND-4.
- **1Y, 2Y, 3Y and 4Y:** Outputs of AND-1, AND-2, AND-3 and AND-4, respectively.
- **GND and Vcc:** Supply connection lines.

7408

**7432 (Quad 2-In OR)**

- **1A and 1B:** Inputs of OR-1.
- **2A and 2B:** Inputs of OR-2.
- **3A and 3B:** Inputs of OR-3.
- **4A and 4B:** Inputs of OR-4.
- **1Y, 2Y, 3Y and 4Y:** Outputs of OR-1, OR-2, OR-3 and OR-4, respectively.
- **GND and Vcc:** Supply connection lines.

## LAB TASKS:

**TASK-I: Analyze AND, OR and NOT ICs using Logic Trainer**

Analyze AND, OR and NOT ICs using logic trainer and record your observations below.

| A | B | A’ | B’ | A + B | AB |
|---|---|---|---|---|---|
| 0 | 0 | | | | |
| 0 | 1 | | | | |
| 1 | 0 | | | | |
| 1 | 1 | | | | |

**TASK-II: Analyze AND, OR and NOT Gates in EWB**

Analyze AND, OR and NOT gates (Logic Gates toolbox) by connecting their outputs to Red Probes (Indicators toolbox) and inputs to Ground and +Vcc Voltage source (Sources toolbox) independently and state your observations below.

**Observations:**

## TASK-III: Find the Truth Table of a Gate using the Logic Converter

The Logic Converter can be found in the ***Instruments*** toolbox. It is used to derive a truth table from circuit schematic.

- Attach the inputs of the logic gate to the input terminals of the logic converter. There are eight input terminals on the logic converter. Use as per your need.
- Connect the output of the circuit to the single output terminal of the logic converter.
- Double click on the logic converter, to open the logic converter display.
- Click on the circuit to truth table button.
- The truth table for the circuit will appear in the logic converter's display. The logic converter tries all possible combinations of the circuit inputs and derive its truth table.

In the figure below, a two input AND gate is connected to the logic converter and its truth table is obtained. Try the same for OR and NOT gates and obtain their respective truth tables.

**<u>TASK-IV: Find the Truth Table for the Circuit:</u>**

Using logic converter, find the truth table for the circuit below and fill the table.

| A | B | A+B | (A+B)’ |
|---|---|---|---|
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

**<u>Note that, logic 1 means connecting the input to +Vcc Voltage Source and logic 0 means connecting the input to Ground.</u>**

## TASK-V: Find the Truth Table of a Three Input Gate:

In EWB, the inputs of a gate can be changed through the following steps.

- Drag a two input gate onto the workspace in EWB.
- Double click the gate. The gate Properties dialogue box will appear.
- Click on the Number of Inputs tab and choose your desired inputs.

Using logic converter, derive the truth tables for three inputs AND and three inputs OR gates, and fill the tables below.

| Inputs | | | Output |
|---|---|---|---|
| **A** | **B** | **C** | **ABC** |
| 0 | 0 | 0 | |
| 0 | 0 | 1 | |
| 0 | 1 | 0 | |
| 0 | 1 | 1 | |
| 1 | 0 | 0 | |
| 1 | 0 | 1 | |
| 1 | 1 | 0 | |
| 1 | 1 | 1 | |

Truth Table for a three-input AND gate

| Inputs | | | Output |
|---|---|---|---|
| **A** | **B** | **C** | **A+B+C** |
| 0 | 0 | 0 | |
| 0 | 0 | 1 | |
| 0 | 1 | 0 | |
| 0 | 1 | 1 | |
| 1 | 0 | 0 | |
| 1 | 0 | 1 | |
| 1 | 1 | 0 | |
| 1 | 1 | 1 | |

Truth Table for a three-input OR gate

**TASK-VI: Find the Truth Table for the Circuit:**

Using logic converter, find the truth table for the circuit below.

| A | B | C | D | X |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 1 | |
| 0 | 0 | 1 | 0 | |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 0 | 1 | |
| 0 | 1 | 1 | 0 | |
| 0 | 1 | 1 | 1 | |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 1 | |
| 1 | 0 | 1 | 0 | |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 0 | |
| 1 | 1 | 0 | 1 | |
| 1 | 1 | 1 | 0 | |
| 1 | 1 | 1 | 1 | |

# LAB-III

## BOOLEAN ANALYSIS OF LOGIC CIRCUITS

### OBJECTIVES:

- To convert logic circuits to Boolean expressions.
- To convert Boolean expressions to logic circuits.
- To determine truth tables for logic circuits using EWB and Logic Trainer.

### INTRODUCTION:

Logic circuits are constructed from a combination of different logic gates. Boolean algebra is the mathematics of digital circuits. The operation of logic circuit can be expressed in a concise manner through Boolean algebra.

Logic can be represented in three ways.

1. Boolean Expressions
2. Boolean Circuits
3. Truth Tables

It is therefore, important to know the inter-conversion between them.

**Converting Logic Circuits to Boolean Expressions:**

To determine Boolean expression of any logic circuit, start at the left most inputs and work towards the final output, writing logic expressions at the output of each gate, as shown below.

## Converting Boolean Expressions to Logic Circuits:

To convert a Boolean expression to logic circuit, requires the knowledge of order of operation. The order of operation is as follow.

1. Bracketed quantities
2. NOTs

3. ANDs
4. ORs

Below example shows step by step conversion of Boolean expression ***F = (A + BC)(A' + B)*** into logic circuit.

**F = (A+BC)(A'+B)**

**F = (A+P)(A'+B)**

**F = Q(A'+B)**

**F = Q(R+B)**

**F = QS**

## LAB TASKS:

### TASK-I: Using AND, OR and NOT ICs

Use AND, OR and NOT ICs from the previous lab to implement the circuit below, using logic trainer. Record your observations in the table below and write the Boolean expression for the circuit.

| A | B | C | D |
|---|---|---|---|
| 0 | 0 | 0 | |
| 0 | 0 | 1 | |
| 0 | 1 | 0 | |
| 0 | 1 | 1 | |
| 1 | 0 | 0 | |
| 1 | 0 | 1 | |
| 1 | 1 | 0 | |
| 1 | 1 | 1 | |

**Boolean Expression: _____**

## TASK-II: Finding Boolean Expressions of Logic Circuits:

Draw the circuits below on EWB, run them to fill in their respective truth tables and find the Boolean expressions of the circuits using logic converter.

**Boolean Expression**

**Logic Circuit – I:**

| X | Y | O |
|---|---|---|
| 0 | 0 | |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | |

**Boolean Expression:** _____

**Logic Circuit – II**

Note: First find the truth table and Boolean function for output X and then for output Y.

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

**Boolean Expression for X: _____**

**Boolean Expression for Y: _____**

# LAB-IV

## BOOLEAN ALGEBRA AND SIMPLIFICATION OF BOOLEAN EXPRESSIONS

### OBJECTIVES:

- To verify the rules, laws and DeMorgan's theorems of Boolean algebra using EWB and AM-2000 logic trainer.
- To simplify Boolean expressions using EWB.
- To compare original Boolean expressions with the simplified versions and to prove that they are equal using EWB and logic trainer.

### INTRODUCTION:

**Rules of Boolean Algebra:**

| | |
|---|---|
| Rule 1 | A + 0 = A |
| Rule 2 | A + 1 = 1 |
| Rule 3 | A . 0 = 0 |
| Rule 4 | A . 1 = A |
| Rule 5 | A + A = A |
| Rule 6 | A + A' = 1 |
| Rule 7 | A . A = A |
| Rule 8 | A . A' = 0 |
| Rule 9 | A'' = A |
| Rule 10 | A + AB = A |
| Rule 11 | A + A'B = A+B |
| Rule 12 | (A + B) (A + C) = A + BC |

## Laws of Boolean Algebra:

**Commutative Laws:**

Commutative Law of Addition:

$$A + B = B + A$$

Commutative Law of Multiplication:

$$A . B = B . A$$

**Associative Laws:**

Associative Law of Addition:

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

Associative Law of Multiplication:

$$A(BC) = (AB)C$$

**Distributive Law:**

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

## DeMorgan's Theorems:

**Theorem-I:** $(AB)' = A' + B'$

**Theorem-II:** $(A + B)' = A'B'$

## Simplification of Boolean Expressions:

Simplification results in an expression with the least amount of terms having the least amount of variables in either complemented or un-complemented forms. A simplified Boolean expression results in the same output as that of the original expression but uses fewer resources (logic gates). Thus, a simplified expression is lot more efficient and less costly to implement.

## LAB TASKS:

### TASK-I: Verify the Rules of Boolean Algebra:

Verify the rules of Boolean algebra in EWB and logic trainer using AND, OR and NOT ICs. Fill the tables below and state your observations.

Rule 1: A + 0 = A

| **A** | **0** | **A + 0** |
|---|---|---|
| 0 | 0 | |
| 1 | 0 | |

Rule 2: A + 1 = 1

| **A** | **1** | **A + 1** |
|---|---|---|
| 0 | 1 | |
| 1 | 1 | |

Rule 3: A . 0 = 0

| **A** | **0** | **A . 0** |
|---|---|---|
| 0 | 0 | |
| 1 | 0 | |

Rule 4: A . 1 = A

| **A** | **1** | **A . 1** |
|---|---|---|
| 0 | 1 | |
| 1 | 1 | |

Rule 5: A + A = A

| **A** | **A** | **A + A** |
|---|---|---|
| 0 | 0 | |
| 1 | 1 | |

Rule 6: A + A’ = 1

| **A** | **A’** | **A + A’** |
|---|---|---|
| 0 | | |
| 1 | | |

Rule 7: A . A = A

| A | A | A . A |
|---|---|---|
| 0 | 0 | |
| 1 | 1 | |

Rule 8: A . A’ = 0

| A | A’ | A . A’ |
|---|---|---|
| 0 | | |
| 1 | | |

Rule 9: A’’ = A

| A | A’ | A’’ |
|---|---|---|
| 0 | | |
| 1 | | |

Rule 10: A + AB = A

| A | B | AB | A + AB |
|---|---|---|---|
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

Rule 11: A + A’B = A + B

| A | B | A’ | A’B | A+B | A+A’B |
|---|---|---|---|---|---|
| 0 | 0 | | | | |
| 0 | 1 | | | | |
| 1 | 0 | | | | |
| 1 | 1 | | | | |

Rule 12: (A+B)(A+C) = A + BC

| A | B | C | A+B | A+C | BC | A+ BC | (A+B) (A+C) |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | | | | | |
| 0 | 0 | 1 | | | | | |
| 0 | 1 | 0 | | | | | |
| 0 | 1 | 1 | | | | | |
| 1 | 0 | 0 | | | | | |
| 1 | 0 | 1 | | | | | |
| 1 | 1 | 0 | | | | | |
| 1 | 1 | 1 | | | | | |

<u>Observations:</u>

**TASK-II: Verify the Laws of Boolean Algebra:**

Verify the laws of Boolean algebra in EWB and logic trainer using AND, OR and NOT ICs. Fill the tables below.

Law 1.1: A + B = B + A

| A | B | A+B | B+A |
|---|---|---|---|
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

Law 1.2: A . B = B . A

| A | B | AB | BA |
|---|---|---|---|
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

Law 2.1: A + (B + C) = (A + B) + C

| A | B | C | A+B | B+C | A+(B+C) | (A+B)+C |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | | | | |
| 0 | 0 | 1 | | | | |
| 0 | 1 | 0 | | | | |
| 0 | 1 | 1 | | | | |
| 1 | 0 | 0 | | | | |
| 1 | 0 | 1 | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | | | | |
| 1 | 1 | 1 | | | | |

Law 2.2: A(BC) = (AB)C

| **A** | **B** | **C** | **AB** | **BC** | **A(BC)** | **(AB)C** |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | | | | |
| 0 | 0 | 1 | | | | |
| 0 | 1 | 0 | | | | |
| 0 | 1 | 1 | | | | |
| 1 | 0 | 0 | | | | |
| 1 | 0 | 1 | | | | |
| 1 | 1 | 0 | | | | |
| 1 | 1 | 1 | | | | |

Law 3: A(B + C) = AB + AC

| **A** | **B** | **C** | **B+C** | **AB** | **AC** | **A(B+C)** | **AB+AC** |
|---|---|---|---|---|---|---|---|
| **0** | **0** | **0** | | | | | |
| **0** | **0** | **1** | | | | | |
| **0** | **1** | **0** | | | | | |
| **0** | **1** | **1** | | | | | |
| **1** | **0** | **0** | | | | | |
| **1** | **0** | **1** | | | | | |
| **1** | **1** | **0** | | | | | |
| **1** | **1** | **1** | | | | | |

## TASK-III: Verify DeMorgan's Theorems:

Verify DeMorgan's theorems in EWB and logic trainer using AND, OR and NOT ICs. Fill the tables below.

Theorem I: (AB)' = A' + B'

| A | B | A' | B' | AB | (AB)' | A'+B' |
|---|---|---|---|---|---|---|
| 0 | 0 | | | | | |
| 0 | 1 | | | | | |
| 1 | 0 | | | | | |
| 1 | 1 | | | | | |

Theorem II: (A + B)' = A'B'

| A | B | A' | B' | A+B | (A+B)' | A'B' |
|---|---|---|---|---|---|---|
| 0 | 0 | | | | | |
| 0 | 1 | | | | | |
| 1 | 0 | | | | | |
| 1 | 1 | | | | | |

.

## TASK-IV: Simplification using EWB:

Simplify the Boolean expression below, using EWB and draw its simplified circuit.

**A(BC + B'C) + AC' + BC' + AB'C'**

Follow the following steps:

1. Enter the Boolean expression inside the marked space on the logic converter, shown in the figure on the right.
2. Press to obtain its truth table.
3. To obtain simplified Boolean expression, press

4. To draw the simplified circuit, press


Logic Circuit:

# LAB-V

## UNIVERSAL GATES

### OBJECTIVES:

- To construct AND, OR, NOT and NOR gates using only NAND gate(s).
- To construct AND, OR, NOT and NAND gates using only NOR gate(s).
- To implement a Boolean function using only NAND gates in EWB and logic trainer.
- To implement a Boolean function using only NOR gates in EWB and logic trainer.
- To implement Boolean function with only NAND gates, using the EWB's logic converter feature.

### INTRODUCTION:

**Universal Gates:**

NAND and NOR are referred to as universal gates, because individually both these gates can implement any Boolean function without the need to use any other type of logic gate.

NAND and NOR are both economical and easier to fabricate, compared to the rest. Because of this reason even an AND gate is typically implemented as a NAND gate followed by a NOT gate, and not the other way around. Similarly an OR gate is typically implemented as a NOR gate followed by a NOT gate, and not the other way around.

**NAND Gate:**

A NAND gate has more than one input and one output. Logic symbol of a two inputs NAND gate is shown in the figure below.

If the two inputs are A and B, the Boolean expression for a NAND gate will be (AB)'.

The output of a NAND gate is LOW, if and only if all the inputs are HIGH, else LOW. Truth table for a two input NAND gate is given below.

| A | B | (AB)' |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

According to DeMorgan's theorem:

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

Hence, both NAND and Negative OR logic gates are equivalent.

## Construction of NOT, AND, OR and NOR gates from NAND gate(s):

## NOR Gate:

A NOR gate has more than one input and one output. Logic symbol of a two inputs NOR gate is shown in the figure below.

If the two inputs are A and B, the Boolean expression for a NOR gate will be (A+B)'.

The output of a NOR gate is HIGH, if and only if all the inputs are LOW, else HIGH. Truth table for a two input NOR gate is given below.

| A | B | (A+B)’ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

According to DeMorgan’s theorem:

**(A+B)’ = A’B’**

Hence, both NOR and Negative AND logic gates are equivalent.

**<u>Construction of NOT, OR, AND and NAND gates from NOR gate(s):</u>**

**Implementation of a Two Level Boolean Function with only NAND gates:**

Follow the steps below, to implement a two level Boolean function with only NAND gates.

1. Simplify the Boolean function and express it in sum of products form.
2. Draw a NAND gate, for each product term.
3. Draw a NAND gate, with inputs coming from the outputs of step-2 NAND gates.

**Implementation of a Two Level Boolean Function with only OR gates:**

Follow the steps below, to implement a two level Boolean function with only NOR gates.

1. Simplify the Boolean function and express it in product of sums form.
2. Draw a NOR gate, for each sum term.
3. Draw a NOR gate, with inputs coming from the outputs of step-2 NOR gates.

## Pin Configuration of NAND and NOR ICs:

**7400 (Quad 2-In NAND)**

- **1A and 1B:** Inputs of NAND-1.
- **2A and 2B:** Inputs of NAND-2.
- **3A and 3B:** Inputs of NAND-3.
- **4A and 4B:** Inputs of NAND-4.
- **1Y, 2Y, 3Y and 4Y:** Outputs of NAND-1, NAND-2, NAND-3 and NAND-4, respectively.
- **GND and Vcc:** Supply connection lines.

**7408 (Quad 2-In AND)**

- **1A and 1B:** Inputs of NOR-1.
- **2A and 2B:** Inputs of NOR-2.
- **3A and 3B:** Inputs of NOR-3.
- **4A and 4B:** Inputs of NOR-4.
- **1Y, 2Y, 3Y and 4Y:** Outputs of NOR-1, NOR-2, NOR-3 and NOR-4, respectively.
- **GND and Vcc:** Supply connection lines.

## LAB TASKS:

**TASK-I: Construct NOT, AND, OR and NOR with NAND gates**

Construct NOT, AND, OR and NOR gates with only NAND gates on logic trainer and fill the tables below.

NOT

| A | (A.A)' |
|---|---|
| 0 | |
| 1 | |

AND

| A | B | (AB)' | (AB)'' |
|---|---|---|---|
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

OR

| A | B | A’ | B’ | (A’B’)’ |
|---|---|---|---|---|
| 0 | 0 | | | |
| 0 | 1 | | | |
| 1 | 0 | | | |
| 1 | 1 | | | |

NOR

| A | B | A’ | B’ | (A’B’)’ | (A‘B’)’’ |
|---|---|---|---|---|---|
| 0 | 0 | | | | |
| 0 | 1 | | | | |
| 1 | 0 | | | | |
| 1 | 1 | | | | |

**TASK-II: Construct NOT, AND, OR and NAND with NOR gates**

Construct NOT, AND, OR and NOR gates with only NOR gates on EWB and fill the tables below.

NOT

| A | (A+A)’ |
|---|---|
| 0 | |
| 1 | |

OR

| A | B | (A+B)’ | (A+B)’’ |
|---|---|---|---|
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

AND

| A | B | A’ | B’ | (A’+B’)’ |
|---|---|---|---|---|
| 0 | 0 | | | |
| 0 | 1 | | | |
| 1 | 0 | | | |
| 1 | 1 | | | |

NAND

| A | B | A’ | B’ | (A’+B’)’ | (A‘+B’)’’ |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

**<u>TASK-III: Implement a Two Level Boolean Function with NAND gates</u>**

Implement the following Boolean function with only NAND gates on EWB and draw the circuit diagram.

$$F = ABC + AC + D$$

Circuit Diagram:

## TASK-IV: Implement a Two Level Boolean Function with NOR gates

Implement the following Boolean function with only NOR gates on logic trainer and draw the circuit diagram.

$$F = ABC + AC + D$$

Circuit Diagram:

## TASK-V: Implement a Boolean Function with NAND gates, using Logic Converter

Implement the function below with NAND gates using the EWB's logic converter and draw the circuit diagram.

$$F = A(B'C + C') + BC' + A$$

Write the Boolean expression in the logic trainer and press button.

**Circuit Diagram:**

# LAB-VI

## ADDERS AND SUBTRACTORS

### OBJECTIVES:

- To design half adder and full adder.
- To design half subtractor and full subtractor.
- To design a parallel binary adder.

### INTRODUCTION:

**Half Adder:**

A half adder accepts two bits at its inputs and produces two outputs, sum and carry-out. Truth table of a half adder is as follow.

**Truth Table:**

| Inputs | | Outputs | |
|---|---|---|---|
| **A** | **B** | **S** | **$C_{out}$** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

**Boolean Functions:**

$$S = A'B + AB' = A \oplus B$$

$$C_{out} = AB$$

## Full Adder:

A full adder accepts three bits at its inputs and produces two outputs, sum and carry-out. Truth table of a full adder is as follow.

**Truth Table:**

| Inputs | | | Outputs | |
|---|---|---|---|---|
| **A** | **B** | **C** | **S** | **$C_{out}$** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

**Boolean Functions:**

$$S = A'B'C + A'BC' + AB'C' + ABC$$

$$S = (A'B' + AB)\,C + (A'B + AB')\,C'$$

$$S = (A \oplus B)'\,C + (A \oplus B)\,C'$$

$$\mathbf{S = (A \oplus B) \oplus C}$$

$$C_{out} = A'BC + AB'C + ABC' + ABC$$

$$C_{out} = (A'B + AB')\,C + AB\,(C' + C)$$

$$\mathbf{C_{out} = (A \oplus B)\,C + AB}$$

## Half Subtractor:

A half subtractor accepts two bits at its inputs and produces two outputs, difference and borrow-in. Truth table of a half subtractor is as follow.

**Truth Table:**

| Inputs | | Outputs | |
|---|---|---|---|
| **A** | **B** | **D** | **$B_{in}$** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

**Boolean Functions:**

$$D = A'B + AB' = A \oplus B$$

$$B_{in} = A'B$$

**<u>Full Subtractor:</u>**

A full subtractor accepts three bits at its inputs and produces two outputs, difference and borrow-in. Truth table of a full subtractor is as follow.

**Truth Table:**

| Inputs | | | <u>Outputs</u> | |
|---|---|---|---|---|
| **A** | **B** | **C** | **D** | **$B_{in}$** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

**Boolean Functions:**

$$D = A'B'C + A'BC' + AB'C' + ABC$$
$$D = (A'B' + AB)\,C + (A'B + AB')\,C'$$
$$D = (A \oplus B)'\,C + (A \oplus B)\,C'$$
$$\mathbf{D = (A \oplus B) \oplus C}$$

$$Bin = A'B'C + A'BC' + A'BC + ABC$$
$$Bin = (A'B' + AB)\,C + A'B\,(C' + C)$$
$$\mathbf{Bin = (A \oplus B)'C + A'B}$$

## Parallel Binary Adder:

Two or more full adders are connected to form a parallel binary adder. In the figure on the right a 2-bit parallel binary adder is drawn by connecting two full adders. The 2-bit parallel binary adder can add two 2-bit numbers.

## Pin Configuration of XOR IC:

### 7486 (Quad 2-In XOR)

- Pin description is self-explanatory.

## LAB TASKS:

### TASK-I: Implement Half Adder

Implement half adder on logic trainer. Use AND and XOR gates and draw the logic circuit below.

**Logic Circuit:**

## TASK-II: Implement Half Adder

Implement full adder on logic trainer. Use NOT, AND, OR and XOR gates and draw the logic circuit below.

**Logic Circuit:**

## TASK-III: Implement Half Subtractor

Implement half subtractor on logic trainer and draw the logic circuit.

**Logic Circuit:**

## TASK-IV: Implement Full Subtractor

Implement full subtractor on logic trainer and draw the logic circuit.

**Logic Circuit:**

## TASK-V: Design a Four Bit Parallel Binary Adder:

Design a four bit parallel binary adder using the digital toolbox of EWB to add two 4-bit binary numbers. Draw the logic diagram below.

**Logic Diagram:**

# LAB-VII

## DECODERS

### OBJECTIVES:

- To understand the operations of 2 x 4, 3 x 8 and 4 x 16 decoders.
- To implement higher order decoders from lower order decoders.
- To implement Boolean functions using decoders.

### INTRODUCTION:

**Decoder:**

A decoder detects the presence of a specific combinations of bits at its inputs and indicates their presence by a specific voltage levels at its outputs. If a decoder has n input lines, it can have 1 to $2^n$ output lines.

Truth table, block diagram and logic diagram of a 2 x 4 decoder (active-LOW outputs) are shown below.

**Truth Table:**

| Inputs | | Outputs | | | |
|---|---|---|---|---|---|
| **A** | **B** | **$D_0$** | **$D_1$** | **$D_2$** | **$D_3$** |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

**Block Diagram:**

**Logic Diagram:**

**Implementing Higher Order Decoder from Lower Order Decoders:**

In the figure below a 3 x 8 decoder is implemented using three 2 x 4 decoders.

- Ground the MSB input of Decoder I.
- The LSB input of Decoder I will act as the MSB for the 3 x 8 decoder. When its 0, output Y0 will be selected, which will enable decoder II, and when its 1, output Y1 will be selected and decoder II will be enabled.
- Connect the LSB input of decoder II with the LSB input of decoder III.
- Connect the MSB input of decoder II with the MSB input of decoder III.

## Boolean Function Implementation using Decoder:

Follow the following procedure to implement a Boolean function using decoder.

- The Boolean function must me in standard SOP form. If it is not, convert it to standard SOP.
- Choose a decoder that has the same inputs as the number of variables in the domain of the Boolean function.
- According to the min terms in the Boolean function, choose the decoder outputs and feed them to a NAND gate if the decoder is active LOW, or OR gate if it is active HIGH.

**Example:**

$$F_{(A,B,C)} = \sum (0,4,6,7)$$

## Pin Configuration of Decoder/DEMUX ICs:

### 74139:

74139 is a dual 2 x 4 Decoder/DEMUX. It has two decoders/DEMUX, one on the either side. Pin description is as follows.

74139

- **1A and 1B:** Input/select lines for decoder/DEMUX 1, where 1A is the LSB and 1B is the MSB.
- **2A and 2B:** Input/select lines for decoder/DEMUX 2, where 2A is the LSB and 2B is the MSB.
- **1Y0 through 1Y3:** Active low data outputs of decoder/DEMUX 1.
- **2Y0 through 2Y3:** Active low data outputs of decoder/DEMUX 2.
- **1G':** Active low enable input for decoder/DEMUX 1.
- **2G':** Active low enable input for decoder/DEMUX 2.
- **VCC and GND:** Supply connection lines.

### 74138:

74138 is a 3 x 8 Decoder IC. Pin description is as follows.

74138

- **A, B and C:** Input/select lines with A being the LSB and C being the MSB.
- **Y0 through Y7:** Active low data outputs.
- **G2A' and G2B':** Active low enable inputs.
- **G1:** Active high enable input.
- **VCC and GND:** Supply connection lines.

**74154:**

74154 is a 4 x 16 Decoder IC. Pin description is as follows.

- **A, B, C and D:** Input/select lines with A being the LSB and D being the MSB.
- **0 through 15:** Active low data outputs.
- **G1' and G2':** Active low enable inputs.
- **VCC and GND:** Supply connection lines.

## LAB TASKS:

**TASK-I: Implement a 3 x 8 Decoder:**

Implement a 3 x 8 decoder on logic trainer and fill the table below.

| Inputs | | | Outputs | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **A** | **B** | **C** | **Y0** | **Y1** | **Y2** | **Y3** | **Y4** | **Y5** | **Y6** | **Y7** |
| 0 | 0 | 0 | | | | | | | | |
| 0 | 0 | 1 | | | | | | | | |
| 0 | 1 | 0 | | | | | | | | |
| 0 | 1 | 1 | | | | | | | | |
| 1 | 0 | 0 | | | | | | | | |
| 1 | 0 | 1 | | | | | | | | |
| 1 | 1 | 0 | | | | | | | | |
| 1 | 1 | 1 | | | | | | | | |

**TASK-II: Design a 3 x 8 decoder from three 2 x 4 decoders:**

On EWB, design a 3 x 8 decoder using three 2 x 4 decoders and fill the table below.

On the toolbar, click on the

digital ICs button. From the toolbox, drag 741xx digital IC on to the workspace and choose your desired IC from the list, as shown below.

| Inputs | | | Outputs | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **A** | **B** | **C** | **Y0** | **Y1** | **Y2** | **Y3** | **Y4** | **Y5** | **Y6** | **Y7** |
| 0 | 0 | 0 | | | | | | | | |
| 0 | 0 | 1 | | | | | | | | |
| 0 | 1 | 0 | | | | | | | | |
| 0 | 1 | 1 | | | | | | | | |
| 1 | 0 | 0 | | | | | | | | |
| 1 | 0 | 1 | | | | | | | | |
| 1 | 1 | 0 | | | | | | | | |
| 1 | 1 | 1 | | | | | | | | |

## TASK-III: Implement Full adder using Decoder and NAND gate:

Implement full adder using a decoder and NAND gate on EWB and draw the logic diagram below.

**Logic Diagram:**

# LAB-VIII

## MULTIPLEXERS

### OBJECTIVES:

- To understand the operation of multiplexers.
- To implement Boolean functions using multiplexer.

### INTRODUCTION:

**Multiplexer:**

A multiplexer has multiple inputs, single output and data select lines. The number of data select lines depends on the number of input lines. For $\mathbf{2^n}$ input lines, we require **n** data select lines. The data select lines decide which input's line data should be transmitted on the single output line, as shown in the table below. In the block diagram, S0 and S1 are data select lines, A, B, C and D are data input lines and Y is the output line.

**Block Diagram:**

**Table:**

| Data Sect Lines | | Output Line |
|---|---|---|
| $S_1$ | $S_0$ | Y |
| 0 | 0 | A |
| 0 | 1 | B |
| 1 | 0 | C |
| 1 | 1 | D |

**Output Function:**

The output function for 4 x 1 multiplexer is,

$$Y = AS_1'S_0' + BS_1'S_0 + CS_1S_0' + DS_1S_0$$

which can be implemented as below.

**Logic Diagram:**

## Boolean Function Implementation:

To implement Boolean function using multiplexer, follow the steps below.

- Express the Boolean function in standard SOP form.
- Choose a multiplexer that has one less data select line than the number of variables in the domain of the Boolean function.
- Construct a truth table for the Boolean function.
- Connect the most significant variables of the Boolean function with the data select lines of the multiplexer.

- Express the output as a function of the least significant variable, for every unique combination of other variables in the truth table, and apply them to the data input lines of multiplexer.

**Example:**

$$F(A,B,C) = \sum(1,2,6,7)$$

| A | B | C | F | F in terms of C |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | 1 | F = C |
| 0 | 1 | 0 | 1 | |
| 0 | 1 | 1 | 0 | F = C’ |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 0 | F = 0 |
| 1 | 1 | 0 | 1 | |
| 1 | 1 | 1 | 1 | F = 1 |

Truth Table of the Boolean function

**Implementation of Boolean Function**

## Pin Configuration of Multiplexer ICs:

### 74153:

74139 is a dual 4 x 1 Multiplexer IC. It has two 4 x 1 Multiplexers, one on the either side. Pin description is as follows.

- **1C0, 1C1, 1C2 and 1C3:** Data input lines for MUX-I.
- **2C0, 2C1, 2C2 and 2C3:** Data input lines for MUX-II.
- **1Y:** Output for MUX-I.
- **2Y:** Output for MUX-II.
- **A and B:** Data select lines, with A being the LSB.
- **1G':** Active low enable input for MUX-I.
- **2G':** Active low enable input for MUX-II.
- **VCC and GND:** Supply connection lines.

**74151:**

74151is a 8 x 1 Multiplexer IC. Pin description is as follows.

- **A, B and C:** Data select lines with A being the LSB.
- **D0 through D7:** Data input lines.
- **Y:** Active high output.
- **W:** Active low output.
- **G':** Active low enable input.
- **VCC and GND:** Supply connection lines.

74151

**74154:**

74150 is a 16 x 1 Multiplexer IC. Pin description is as follows.

- **A, B, C and D:** Data select lines with A being the LSB.
- **E0 through E15:** Data input lines.
- **G1':** Active low enable input.
- **W:** Active low output.
- **VCC and GND:** Supply connection lines.

74150

## LAB TASKS:

**TASK-I: Implement the Boolean Function:**

Implement the Boolean function below, using appropriate multiplexer IC, on logic trainer and fill the table.

$$F(A,B,C) = \sum(0, 1, 2, 7)$$

| A | B | C | F | F in terms of C |
|---|---|---|---|---|
| 0 | 0 | 0 | | |
| 0 | 0 | 1 | | F = |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | F = |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | F = |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | F = |

**<u>TASK-II: Implement the Boolean Function:</u>**

Implement the Boolean function below, using appropriate multiplexer IC, on EWB and fill the table.

$$F(A,B,C) = \sum(1, 3, 6, 7)$$

| A | B | C | F | F in terms of C |
|---|---|---|---|---|
| 0 | 0 | 0 | | |
| 0 | 0 | 1 | | F = |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | F = |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | F = |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | F = |

**TASK-III: Implement the Boolean Function:**

Implement the Boolean function below, using appropriate multiplexer IC, on EWB and fill the table and draw the logic diagram.

**F(A,B,C,D) = ∑(1, 5, 9, 10, 11, 14)**

| A | B | C | D | F | F in terms of C |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | |
| 0 | 0 | 0 | 1 | | F = |
| 0 | 0 | 1 | 0 | | |
| 0 | 0 | 1 | 1 | | F = |
| 0 | 1 | 0 | 0 | | |
| 0 | 1 | 0 | 1 | | F = |
| 0 | 1 | 1 | 0 | | |
| 0 | 1 | 1 | 1 | | F = |
| 1 | 0 | 0 | 0 | | |
| 1 | 0 | 0 | 1 | | F = |
| 1 | 0 | 1 | 0 | | |
| 1 | 0 | 1 | 1 | | F = |
| 1 | 1 | 0 | 0 | | |
| 1 | 1 | 0 | 1 | | F = |
| 1 | 1 | 1 | 0 | | |
| 1 | 1 | 1 | 1 | | F = |

**Logic Diagram:**

# LAB-IX

## FLIP-FLOPS

### OBJECTIVES:

- To understand the operations of different flip-flops.
- To convert one type of flip-flop into another.

### INTRODUCTION:

**Flip-Flop:**

Flip-flop is the memory unit or storage element of synchronous sequential circuit. It can store one bit of information, either 1 or 0 at a time. A 1 means that the sequential circuit is in the SET state and a 0 means that it is in the RESET state.

A flip-flop can be either positive edge triggered or negative edge triggered. A positive edge triggered flip-flop changes state only, when the clock pulse goes from LOW to HIGH. A negative edge triggered flip-flop changes state only, when the clock pulse goes from HIGH to LOW.

The next state ($\mathbf{Q_{t+1}}$) of flip-flop, is not only defined by the flip-flop inputs but also by the present state ($\mathbf{Q_t}$) of the storage element.

**Different Types of Flip-Flops:**

**S-R Flip-Flop:**

S-R flip-flop has two inputs, S and R and two outputs Q and Q'. Q is the normal output and Q' is the complemented output. The characteristic table and characteristic equation of S-R flip-flop is given below.

**Characteristic Table:**

| S | R | $Q_t$ | $Q_{t+1}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | x |
| 1 | 1 | 1 | x |

**Characteristic Equation:**

$$Q_{t+1} = S + R'Q_t$$

**D Flip-Flop:**

D flip-flop has one input, D and two outputs Q and Q'. Q is the normal output and Q' is the complemented output. The characteristic table and characteristic equation of D flip-flop is given below.

**Characteristic Table:**

| D | $Q_t$ | $Q_{t+1}$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**Characteristic Equation:**

$$Q_{t+1} = D$$

**J-K Flip-Flop:**

J-K flip-flop has two inputs, J and K and two outputs Q and Q'. Q is the normal output and Q' is the complemented output. The characteristic table and characteristic equation of J-K flip-flop is given below.

**Characteristic Table:**

| J | K | $Q_t$ | $Q_{t+1}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

**Characteristic Equation:**

$$Q_{t+1} = JQ_{t'} + K'Q_t$$

**D Flip-Flop:**

T flip-flop has one input, T and two outputs Q and Q'. Q is the normal output and Q' is the complemented output. The characteristic table and characteristic equation of T flip-flop is given below.

**Characteristic Table:**

| T | $Q_t$ | $Q_{t+1}$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Characteristic Equation:**

$$Q_{t+1} = T \oplus Q_t$$

## Conversions of Flip-Flops:

Using a conversion table, one flip-flop can be converted into another.

A conversion table has three main columns.

- The input(s) of the flip-flop to which the original flip-flop needs to be converted, constitutes the first main column.
- The outputs, both the present state Qt and next state Qt+1, constitutes the second main column.
- The input(s) of the original flip-flop constitutes the third main column.

The number of inputs in the first column along with the present state Qt, defines the total number of possible combinations in the conversion table.

For each combination, find the next state Qt+1 values and then using the information in the second main column fill the third main column.

Finally, find the simplified Boolean function(s) for each entity in the third main column and implement it.

**Example: Conversion of S-R flip-flop to J-K flip flop**

**Conversion Table:**

| **Inputs of J-K Flip-Flop** | | **Outputs** | | **Inputs of S-R Flip-Flop** | |
|---|---|---|---|---|---|
| **J** | **K** | **Qt** | **Qt+1** | **S** | **R** |
| 0 | 0 | 0 | 0 | 0 | X |
| 0 | 0 | 1 | 1 | X | 0 |
| 0 | 1 | 0 | 0 | 0 | X |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | X | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |

**Obtaining Boolean Functions:**

| J \ KQt | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | X | 0 | 0 |
| 1 | 1 | X | 0 | 1 |

S = J Qt'

| J \ KQt | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | X | 0 | 1 | X |
| 1 | 0 | 0 | 1 | 0 |

R = K Qt

**Logic Diagram:**

## Pin Configurations of Different Flip-Flop ICs:

Pin description is self-explanatory.

**7473 (Dual JK FF (clr))**

7473

**7474 (Dual D type FF (pre, clr))**

7474

**7476 (Dual JK FF (pre, clr))**

**74107 (Dual JK FF (clr))**

**74109 (Dual JK FF (+edge, pre, clr))**

**74112 (Dual JK FF (-edge, pre, clr))**

## LAB TASKS:

**TASK-I: Use 74109 (Dual JK FF (+edge, pre, clr)) IC to fill the table below:**

| J | K | Clk | Q |
|---|---|---|---|
| 0 | 0 | 0 | |
| 0 | 0 | 1 | |
| 0 | 1 | 0 | |
| 0 | 1 | 1 | |
| 1 | 0 | 0 | |
| 1 | 0 | 1 | |

| | | | |
|---|---|---|---|
| 1 | 1 | 0 | |
| 1 | 1 | 1 | |

**TASK-II: Use 74112 (Dual JK FF (+edge, pre, clr)) IC to fill the table below:**

| J | K | Clk | Q |
|---|---|---|---|
| 0 | 0 | 0 | |
| 0 | 0 | 1 | |
| 0 | 1 | 0 | |
| 0 | 1 | 1 | |
| 1 | 0 | 0 | |
| 1 | 0 | 1 | |
| 1 | 1 | 0 | |
| 1 | 1 | 1 | |

**TASK-III: Convert J-K to S-R**

Convert J-K flip-flop to SR flip-flop. Fill the conversion table below. Write Boolean functions for J and K and draw the logic diagram.

**Conversion Table:**

| Inputs of S-R Flip-Flop | | Outputs | | Inputs of J-K Flip-Flop | |
|---|---|---|---|---|---|
| J | K | Qt | Qt+1 | S | R |
| 0 | 0 | 0 | | | |
| 0 | 0 | 1 | | | |
| 0 | 1 | 0 | | | |
| 0 | 1 | 1 | | | |
| 1 | 0 | 0 | | | |

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 1 | | | |
| 1 | 1 | 0 | | | |
| 1 | 1 | 1 | | | |

**Boolean Functions:**

**Logic Diagram:**

## TASK-IV: Convert D to J-K

Convert D flip-flop to J-K flip-flop, and draw the logic diagram below.

**Logic Diagram:**

# LAB-X

## SYNCHRONOUS SEQUENTIAL CIRCUITS

### OBJECTIVES:

- To analyze synchronous sequential circuits.
- To design synchronous sequential circuits.

### INTRODUCTION:

**Synchronous Sequential Circuits:**

Sequential circuits that are synchronized with a clock are referred to as synchronous or clocked sequential circuits. They are characterized by inputs, outputs and states. Outputs and next states are both, functions of inputs and present states.

The memory or storage element used in a synchronous sequential circuit is a flip-flop.

**Analysis of Synchronous Sequential Circuits:**

There are three ways to analyze a synchronous sequential circuit.

- Algebraic Analysis.
- Tabular Analysis.
- Graphical Analysis.

**Algebraic Analysis:**

Algebraic analysis of synchronous sequential circuit is performed using State Equations. A state equation defines the next state as a function of both the inputs and present states.

**Tabular Analysis:**

Tabular analysis of synchronous sequential circuit is performed using State Table. A state table is similar to a truth table. The input portion of state table consists of external inputs to the sequential circuit and present states. Next states and circuit outputs comprise the output portion.

There are four main columns in a state table, present states, inputs, next states and outputs. The number of rows in a state table depend on the combined total number of external inputs and present states.

**Graphical Analysis:**

Graphical analysis of synchronous sequential circuit is performed using State Diagrams. In a state diagram, a state is represented by a circle and the transition from one state to another is indicated by an arrow. Inputs and outputs are listed over the arrow in the form **(x/y)**, where x is an input and y is an output. Multiple inputs and multiple outputs are separated by **commas.**

**Flip-Flop Input Function:** It is the Boolean function that represent input to a flip-flop, in terms of present states and external input(s).

**Circuit Output Function:** It is the Boolean function that represent output of the sequential circuit, in terms of present states and external input(s).

**<u>Analysis Procedure:</u>**

- Find the flip-flop inputs and circuit output functions
- Obtain state equations, by substituting flip-flop input functions into flip-flop characteristic equations.
- Use the state equations and circuit output functions to fill the next sate and output columns of state table.
- Graphically represent state table by means of state diagrams.

**Design Procedure:**

- Obtain state table from the word description or state diagram, whichever is provided.
- Reduce the sate table.
- Assign binary values to each state.
- Derive a circuit excitation and output table.
- Find the minimized flip-flop input and circuit output functions.
- Draw the logic diagram.

## LAB TASKS:

**TASK-I: Analyze a Synchronous Sequential Circuit**

Analyze the synchronous sequential circuit below, by first implementing it on logic trainer and then find its state equations, state table and state diagram.

**Flip-Flop Input Functions:**

$J_A$ = _____

$K_A$ = _____

$J_B$ = _____

$K_B$ = _____

**Circuit Output Function:**

Y = _____

**State Equations:**

$A_{t+1}$ = _____

_____

_____

$B_{t+1}$ = _____

_____

_____

## State Table:

| Present States | | Input | Next States | | Output |
|---|---|---|---|---|---|
| A | B | X | A | B | Y |
| 0 | 0 | 0 | | | |
| 0 | 0 | 1 | | | |
| 0 | 1 | 0 | | | |
| 0 | 1 | 1 | | | |
| 1 | 0 | 0 | | | |
| 1 | 0 | 1 | | | |

| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 0 | | | |
| 1 | 1 | 1 | | | |

**State Diagram:**

## TASK-II: Design a Synchronous Sequential Circuit

Design a synchronous sequential circuit, using J-K flip-flops from the state diagram given below.

**State Table:**

| Present State | Next State x = 0 | Next State x = 1 | Output x = 0 | Output x = 1 |
|---|---|---|---|---|
| a | | | | |
| b | | | | |
| c | | | | |
| d | | | | |
| e | | | | |
| f | | | | |

**Reduced State Table:**

| Present State | Next State x = 0 | Next State x = 1 | Output x = 0 | Output x = 1 |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**State Assignment:**

| Present States | | Next State x = 0 | | Next State x = 1 | | Output x = 0 | Output x = 1 |
|---|---|---|---|---|---|---|---|
| A | B | A | B | A | B | y | y |
| 0 | 0 | | | | | | |
| 0 | 1 | | | | | | |
| 1 | 0 | | | | | | |
| 1 | 1 | | | | | | |

**Circuit Excitation and Output Table:**

| Inputs of Combinational Circuit | | | Next States | | | | Outputs of Combinational Circuit | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Present States | | Inputs | x = 0 | | x = 1 | | FF Inputs | | | | Output |
| A | B | x | A | B | A | B | JA | KA | JB | KB | y |
| 0 | 0 | 0 | | | | | | | | | |
| 0 | 0 | 1 | | | | | | | | | |
| 0 | 1 | 0 | | | | | | | | | |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | | | | | | | | | |
| 1 | 0 | 0 | | | | | | | | | |
| 1 | 0 | 1 | | | | | | | | | |
| 1 | 1 | 0 | | | | | | | | | |
| 1 | 1 | 1 | | | | | | | | | |

**Flip-Flop Input Functions (minimized):**

$J_A$ = _____

$K_A$ = _____

$J_B$ = _____

$K_B$ = _____

**Circuit Output Function:**

Y = _____

**Logic Diagram:**

# LAB-XI

## COUNTERS

### OBJECTIVES:

- To design different types of counters.

### INTRODUCTION:

**Counters:**

Counters are synchronous sequential circuits that are used for counting purposes. The total number of flip-flops in a counter determine the total number of states that the counter goes through. A counter goes through $\mathbf{2^n}$ states, if the total number of flip-flops used are **n**. The arrangement of flip-flops and the combinational logic used in a counter determine the sequence of counting operation.

Counters can be classified as synchronous or asynchronous counters.

**Synchronous Counters:**

In a synchronous counter, all the flip-flops within the counter are synchronized with a single clock pulse.

**Asynchronous Counters:**

In asynchronous counter, flip-flops within the counter have different clock pulses.

Some common terminologies, related with counters are listed below.

**Binary Counter:** A counter that follows the binary sequence of counting.

**Binary UP Counter:** A binary counter, whose count sequence goes from the least significant value to the most significant value.

**Binary DOWN Counter:** A binary counter, whose count sequence goes from the most significant value to the least significant value.

**Binary UP/DOWN Counter:** A binary counter, which is capable of counting both in the UP as well as in the DOWN direction.

**Modulus:** Modulus of a counter is the total number of states that the counter goes through.

**Maximum Modulus:** Maximum modulus is the maximum number of states that a counter can go through.

**Truncated Sequence:** Truncated sequence is the sequence of states of a counter, whose modulus is less than the maximum modulus.

**Decade Counter:** A truncated sequence counter whose modulus is 10 is referred to as a decade counter.

**BCD Counter:** Example of decade counter.

## LAB TASKS:

### TASK-I: Design a Synchronous UP Counter

Design a 3-bit synchronous UP counter, using J-K flip-flops on logic trainer. Fill the table and draw the logic diagram below. In the table $Q_A$ is the LSB and $Q_C$ is the MSB.

| Clock Pulse | $Q_C$ | $Q_B$ | $Q_A$ |
|---|---|---|---|
| 0 | | | |
| 1 | | | |
| 2 | | | |
| 3 | | | |

| | | | |
|---|---|---|---|
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |
| 8 | | | |

**Logic Diagram:**

## TASK-II: Design Asynchronous DOWN Counter

Design a 3-bit asynchronous DOWN counter, using J-K flip-flops on logic trainer. Fill the table and draw the logic diagram below. In the table $Q_A$ is the LSB and $Q_C$ is the MSB.

| Clock Pulse | $Q_C$ | $Q_B$ | $Q_A$ |
|---|---|---|---|
| 0 | | | |
| 1 | | | |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |
| 8 | | | |

**Logic Diagram:**

## TASK-III: Design a Binary Counter

Design a binary counter that goes through the following states, using J-K flip-flops and draw the logic diagram. Use the unused states as don't cares. In the table below, $Q_A$ is the LSB and $Q_C$ is the MSB.

| Clock Pulse | $Q_C$ | $Q_B$ | $Q_A$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 1 |
| 3 | 1 | 0 | 1 |
| 4 | 1 | 1 | 0 |
| 5 | 1 | 1 | 1 |

**Logic Diagram:**

# LAB-XII

## REGISTERS

### OBJECTIVES:

- To design different types of registers.

### INTRODUCTION:

**Registers:**

Registers have two basic functions.

- Data Storage
- Data Movement

The storage capacity of a register depends on the total number of flip-flops used in its design. A register with **n** flip-flops can store a maximum of **n** bits.

Depending on the type of data movement, a register can be classified into;

- Serial-In, Serial-Out Register
- Serial-In, Parallel-Out Register
- Parallel-In, Serial-Out Register
- Parallel-In, Parallel-Out Register
- Bidirectional Shift Register
- Universal Shift Register

**Serial-In, Serial-Out Register:**

In serial-in, serial-out register, the data bits are entered serially (right most bit first) and are taken out serially.

**Serial-In, Parallel-Out Register:**

In serial-in, parallel-out register, the data bits are entered serially (right most bit first) and are taken out simultaneously in a parallel manner.

**Parallel-In, Serial-Out Register:**

In parallel-in, serial-out register, the data bits are entered simultaneously and are taken out in a bit by bit fashion.

**Parallel-In, Parallel-Out Register:**

In parallel-in, parallel-out register data bits are loaded simultaneously as well as taken out simultaneously.

**Bidirectional Shift Register:**

In bidirectional shift register, that data is loaded serially and taken out serially like a serial-in, serial-out register, but the data can move in either direction from left to right or from right to left.

**Universal Shift Register:**

In universal shift register, the data can be loaded both simultaneously or bit by bit. Similarly the data can be taken out in either simultaneous manner or bit by bit fashion. The data bits can also move in either direction that is from left to right or from right to left.

## LAB TASKS:

**TASK-I: Design a three bit serial-in, serial-out register**

Using D flip-flops ICs, design a 3-bit serial-in, serial-out register on logic trainer.

Serially input 0111001 with the right most bit first and record your observations below. Also draw the logic diagram.

| Clock Pulse | $Q_A$ | $Q_B$ | $Q_C$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | | | |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |

**Logic Diagram:**

**TASK-II: Design a three bit serial-in, parallel-out register**

Using D flip-flops ICs, design a 3-bit serial-in, parallel-out register on logic trainer.

Serially input 0111001 with the right most bit first and record your observations on the 1st, 5th and 7th clock pulse below. Also draw the logic diagram.

| Clock Pulse | $Q_A$ | $Q_B$ | $Q_C$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | | | |
| 5 | | | |
| 7 | | | |

**Logic Diagram:**

# LAB-XIII

## SEMESTER PROJECT

### OBJECTIVE:

- To implement the knowledge you gained during the semester about digital logic designs in a creative manner.

### INTRODUCTION:

Group projects reinforce skills that are relevant to both group and individual work. Some of the benefits of group projects are listed below.

- Breaking complex tasks into parts and steps.
- Planning and managing time.
- Refining understanding through discussion and explanation.
- Giving and receiving feedback on performance.
- Developing stronger communication skills.
- Sharing diverse perspectives.

### PROJECT'S TITLE:

| |
|---|
| |

## PROJECT'S INTRODUCTION:

## PROJECT'S LOGIC DIAGRAM:

## CONCLUSIONS:

## NAMES OF GROUP MEMBERS:

More Books!

yes I want morebooks!

Buy your books fast and straightforward online - at one of the world's fastest growing online book stores! Environmentally sound due to Print-on-Demand technologies.

Buy your books online at

**www.get-morebooks.com**

---

Kaufen Sie Ihre Bücher schnell und unkompliziert online – auf einer der am schnellsten wachsenden Buchhandelsplattformen weltweit!
Dank Print-On-Demand umwelt- und ressourcenschonend produziert.

Bücher schneller online kaufen

**www.morebooks.de**

OmniScriptum Marketing DEU GmbH
Bahnhofstr. 28
D - 66111 Saarbrücken
Telefax: +49 681 93 81 567-9

info@omniscriptum.com
www.omniscriptum.com

OMNIScriptum