

GIFT UNIVERSITY GUJRANWALA



LAB MANUAL

DIGITAL LOGIC DESIGN

DEPARTMENT OF ELECTRICAL ENGINEERING

Prepared By:

Engr. Maryam Idress

Updated By:

Engr. Shuja Ahmad Khokhar

Updated on: March 20, 2019

Name: _____

Registration No: _____

Semester: _____

Batch: _____

Digital Logic Designs (LAB)

List of Experiments

S. No.	Experiment Name
1.	Introduction to Laboratory Equipment and Components.
2.	Basic Logic Gates.
3.	Boolean Functions Simplification and Implementation using Boolean algebra.
4.	Derivation of Boolean Function from Truth Table using Minterms, Simplification & Verification.
5.	Universal Gates
6.	Adders and Subtractors
7.	Encoders and Decoders.
8.	Multiplexers and DE Multiplexers. .
9.	Binary Comparator
10.	Introduction to Logisim Software.
11.	Introduction to Latches.
12.	Introduction to Flip-Flops.
13.	Synchronous & Asynchronous Counters.
14.	Shift Register.

INTRODUCTION TO LABORATORY EQUIPMENTS & COMPONENTS

EXPERIMENT NO. 01

Name	Report Marks (05)	Lab Performance (05)	Viva Marks (05)	Total (15)

EXPERIMENT NO. 01

Introduction to Laboratory Equipment and Components

Objectives

- To become familiar with basic laboratory equipment and components.
- To familiarize with the safety and laboratory rules and regulations.

IT- 300 Digital Logic Training Systems

The IT-300 DIGITAL LOGIC TRAINING SYSTEM is a comprehensive and self-contained system suitable for anyone engaged in digital logic experiments. All necessary equipment for digital logic experiments such as power supply, signal generator, switches and displays are installed on the main unit. The 10 modules cover a wide variety of essential topics in the field of digital logic. It is a time and cost saving trainer for both students and engineers interested in developing and testing circuit proto-types.

- Suitable for combinational logic, sequential logic and microprocessor circuits design and experiments
- Comprehensive power, signal supply and testing devices for convenient experiments
- Experiments are expandable and flexible with universal breadboard
- Capable of processing TTL, CMOS, NMOS, PMOS and ECL circuits
- All supply units are equipped with overload protection for safety purpose
- All modules equipped with 8-bit DIP switch for fault simulations

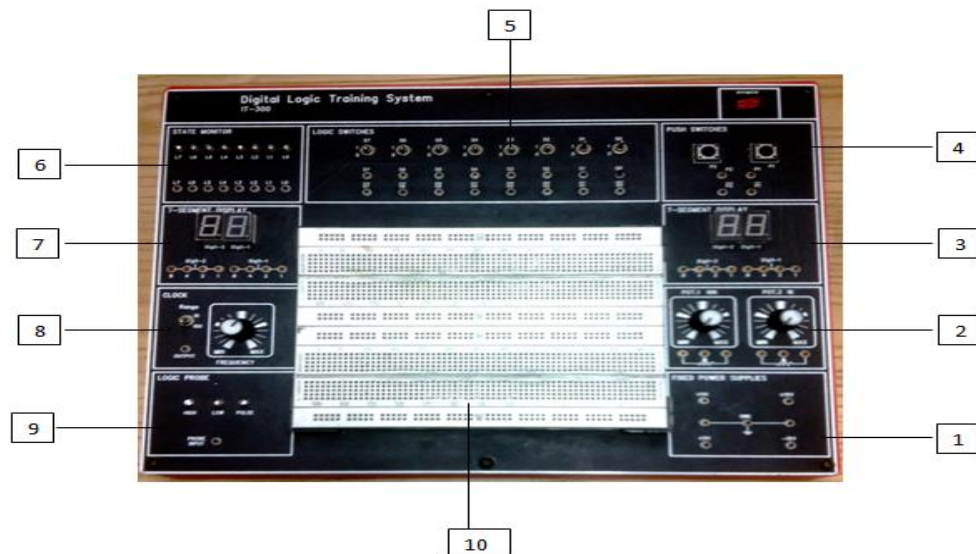


Fig 1.1

Specifications:

1. Fixed Power Supplies

- Voltage Range : +5V, +15V, -15V
- With output overload protection

2. Potentiometers

- Carbon Track, 1K and 10K

3. Seven Segment Display (A)

- Two 7-segment LED displays with BCD to 7-segment decoder/driver

4. Push Switches

- 2 Independent switches, each with Q, Q' output

5. Logic Switch

- 8-bit switch with complimentary TTL outputs

6. State Monitor

- 8 independent LEDs to indicate high and low logic state

7. Seven Segment Display (B)

- Two 7-segment LED displays with BCD to 7-segment decoder/driver

8. Clock

- 1Hz – 64Hz

9. Logic Probe

- “LO” and “HI” LED display low and high state respectively

10. Breadboard

Breadboards are simply a set of pre-wired interconnects that aid you in the building of your circuits. By plugging a wire of one component into a hole you will connect it to all other components in that strip or bus. Using strips, buses, and jumper wires you can construct a circuit on your breadboard. You can tell which holes are connected in one node by the Coloured lines connecting them in Figure 1.2. There are the component connection strips that run up and down and buses labeled with an A or B. After examining Figure 1.2 the use of a breadboard should be clear.

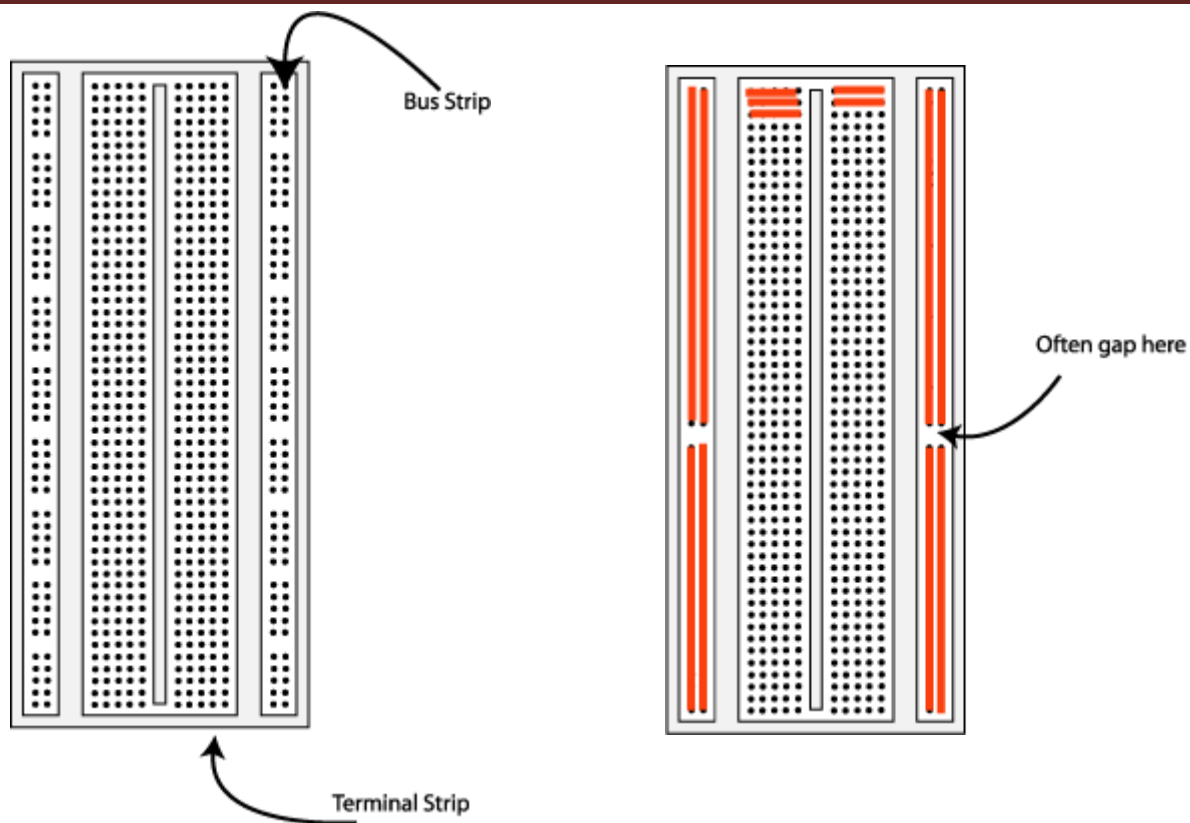


Fig. 1.2 Bread-board connections

1680 tie-point breadboard on top panel can be easily put in and taken off.

- 2 Terminal Strips, Tie-point 1280
- 4 Distribution Strips, Tie-point 400

IC Logic Families and Characteristics

Introduction

- Miniature, low cost electronics circuits whose components are fabricated on a single, continuous piece of semiconductor material to perform a high level function.
- Usually referred to as a monolithic IC
- First introduced in 1958
- Categorized as digital or linear ICs or according to the level of complexity of the IC

Packaging

- Protects the chip from mechanical damage and chemical contamination.
- Provides a complete unit.
- It is large enough for electrical connections to be made

- Material is molded plastic, epoxy, resin, or silicone. Ceramic used if higher thermal dissipation capabilities required. Metal/glass used in special cases.

Three most common packages for IC are

- Dual –in-line (DIP) most common
- Flat pack
- Axial lead

IC Logic Families

Thus far, we have specified the logic level as either 0 or 1, or HIGH or LOW. In circuit implementation, we will have to specify the actual voltage/current levels that constitute a HIGH or a LOW. These standardized voltage/current levels are grouped in families of digital ICs so that ICs belonging to the same family will have the same characteristics.

Common families are

- TTL: Transistor-Transistor Logic
- ECL: Emitter Coupled Logic
- IIL: Integral Injection Logic
- MOS IC: Metal Oxide Semiconductor Integrated Circuits

In our Lab work, we shall use TTL Logic family that corresponds to 74LSxx series.

(LABORATORY REGULATIONS AND SAFETY RULES)

The following Regulations and Safety Rules must be observed in all concerned laboratory location.

1. **It is the duty of all concerned who use any electrical laboratory to take all reasonable steps to safeguard the HEALTH and SAFETY of themselves and all other users and visitors.**
2. **Be sure that all equipment is properly working before using them for laboratory exercises. Any defective equipment must be reported immediately to the Lab. Instructors or Lab. Technical Staff.**
3. **Students are allowed to use only the equipment provided in the experiment manual or equipment used for senior project laboratory.**
4. **Power supply terminals connected to any circuit are only energized with the presence of the Instructor or Lab. Staff.**
5. **Students should keep a safe distance from the circuit breakers, electric circuits or any moving parts during the experiment.**
6. **Avoid any part of your body to be connected to the energized circuit and ground.**
7. **Switch off the equipment and disconnect the power supplies from the circuit before leaving the laboratory.**

8. **Observe cleanliness and proper laboratory housekeeping of the equipment and other related accessories.**
9. **Wear proper clothes and safety gloves or goggles required in working areas that involves fabrications of printed circuit boards, chemicals process control system, antenna communication equipment and laser facility laboratories.**
10. **Double check your circuit connections specifically in handling electrical power machines, AC motors and generators before switching “ON” the power supply.**
11. **Make sure that the last connection to be made in your circuit is the power supply and first thing to be disconnected is also the power supply.**
12. **Equipment should not be removed, transferred to any location without permission from the laboratory staff.**
13. **Software installation in any computer laboratory is not allowed without the permission from the Laboratory Staff.**
14. **Computer games are strictly prohibited in the computer laboratory.**
15. **Students are not allowed to use any equipment without proper orientation and actual hands on equipment operation.**
16. **Smoking and drinking in the laboratory are not permitted.**

All these rules and regulations are necessary precaution in Electrical Laboratory to safeguard the students, laboratory staff, the equipment and other laboratory users.

Observations:

(02 Marks)

Review Questions

1. What is TTL Family?

(01 Mark)

2. What are the various types of logic families?

(01 Mark)

3. What is Breadboard? Why Breadboard is used? Why do we call it a Breadboard?

(01 Mark)

BASIC LOGIC GATES

EXPERIMENT NO. 02

Name	Report Marks (05)	Lab Performance (05)	Viva Marks (05)	Total (15)

EXPERIMENT NO. 02

Basic Logic Gates

Objectives

- Introduction to ICs, logic families, 74xx.
- Familiarization with pins, switches, LEDs and power supply connections.
- Implementation of a basic gate's circuits using ICs.

Components

- IC Type 7400 Quadruple 2-input NAND gates
- IC Type 7402 Quadruple 2-input NOR gates
- IC Type 7404 inverter
- IC Type 7408 Quadruple 2-input AND gates
- IC Type 7432 Quadruple 2-input OR gates
- IC Type 7486 Quadruple 2-input XOR gates
- Wire Stripper
- Prototyping board with power and ground connections

Integrated Circuits and Gates

- In CP-106, you are required to work on digital circuits. Digital circuits are hardware components that are implemented using transistors and interconnections in complex semiconductor devices called *integrated-circuits*. Digital circuits work in binary logic domain which uses two discrete values, **TRUE** (High) and **FALSE (Low)**. We can also refer to these values as **1(High)** and **0 (Low)**.
- Logic-Gates are basic building blocks of digital circuits. Using these building blocks, complex functions or larger digital circuits can be built. Examples of the basic logic gates are **AND, OR, NOT, NAND and NOR**. A complex gate such as an **XOR** gate can be built out of these basic gates. Each logic gate is actually implemented in part of an integrated circuit (IC), with each gate made using several transistors.
- For the most part we do not need to concern ourselves with the actual circuitry inside each gate, only the interconnections between individual gates. Usually each IC package contains several individual gates. The 7408 chip implements 4 two input AND gates and is commonly referred to as a “Quad two input AND” chip. Similarly the 7432 chip is a “Quad 2 input OR” chip while the 7404 chip is a “Hex Inverter” since it contains 6 inverters. The IEEE standard logic symbols for each of these devices are shown in Figure 1.

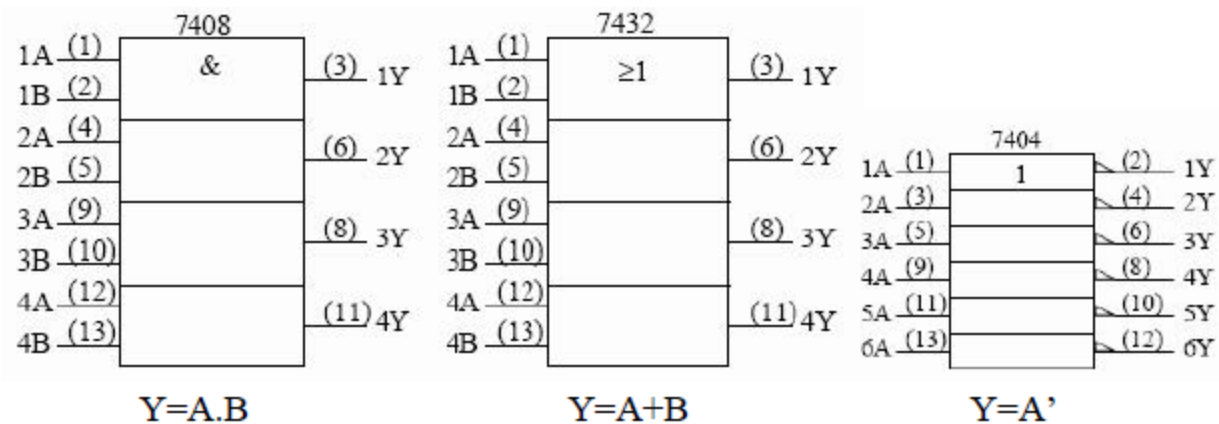


Figure 1: Standard Logic Symbols

Each of the chips in Figure 1 is available in 14 pin Dual-In-Line packages or DIPs. In a popular logic family called TTL (Transistor-Transistor Logic), the low logic level is assigned to 0V and the high logic level is assigned to 5V. Each IC or chip has an ID number that can be referenced in IC Data Book. From the book, you can get the pin configuration of that chip.

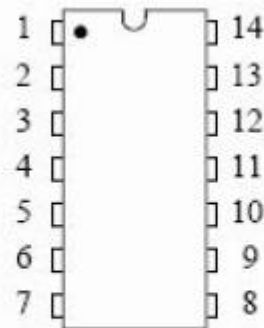


Figure 2: Identifying Pin 1

The pin numbers assigned to each logic signal are shown inside brackets in the figure. The pins are numbered as shown in Figure 2. Pin 1 is usually identified as the pin to the left of an indentation or cutout in one end of the chip that is visible when the chip is viewed from the top. Occasionally, it is also identified by a printed or indented dot placed just next to it.

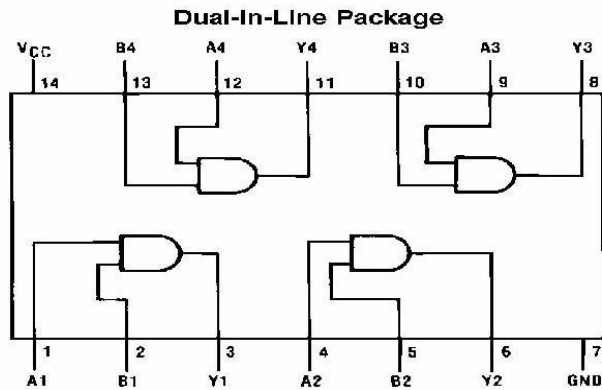


Figure 3: A Quad 2-input AND gate chip

Figure 3, shows an IC which contains four 2-input AND gates. You will notice that pins 7 and 14 show no connections. In 14 pin DIP packages, pin 7 is usually connected to ground (Gnd), and pin 14 is usually connected to the power supply (Vcc). These connections must be made or the chip will not work. Take care not to connect pin 7 to power and pin 14 to ground, or to connect the outputs of two or more gates together. In complex ICs more than one pin can be dedicated for power (VDD) as well as ground. In the simpler gates that we will be using in this experiment, the ICs require only one pin for power (VDD) and another for ground (GND). The power supply (VDD) voltage is typically +5Volts, 3.3 Volts or 2.5 Volts. The ground is typically connected to 0 Volts.

The Prototyping Board

The circuit is constructed on the breadboard section of the prototyping board. A breadboard is used to rapidly create an experimental or prototype circuit. It consists of an array of holes in which wires or component leads can easily be inserted. Rows of five or six holes are electronically connected to form a single node.

When a component lead is inserted into one of the holes, anything inserted into any of the remaining four holes will be connected to that lead. Nodes can be connected to each other using wires with 1/4" of insulation stripped from both ends. The holes are spaced 100 mms (.1 inch) apart, which is the standard spacing of the pins on a DIP package.

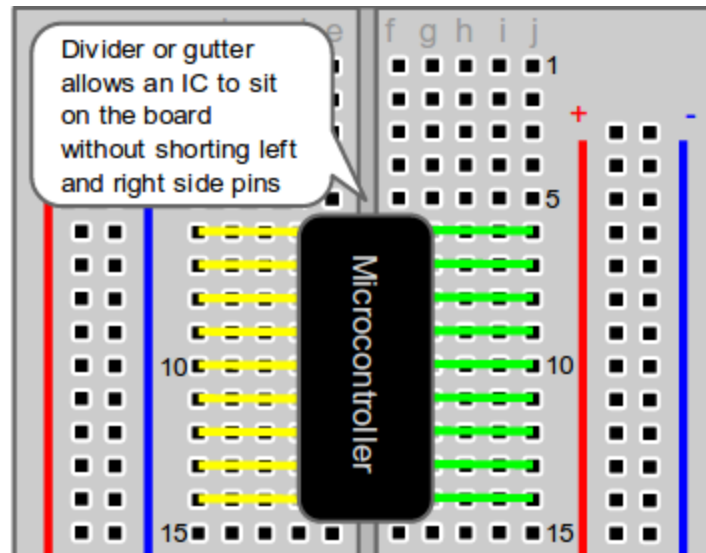


Figure 4: Placing DIP devices on a breadboard

The breadboard has a groove down the center separating one side from the other. When inserting a chip into the breadboard, make sure it fits into the central groove as shown in the figure 4 otherwise the pins on opposite sides of the chip will be connected. Press the chip down until it touches the surface of the breadboard. Devices inserted on the breadboard can be connected to components on the prototyping board by using wires between the device and the J1 connector.

LAB TASK

1. VERIFICATION OF TRUTH TABLE OF AND, OR, NOT, NAND AND NOR LOGIC GATES

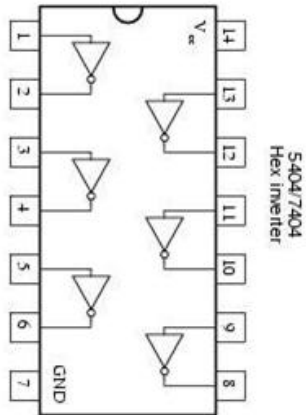
(03 Marks)

- **Hardware Connections**

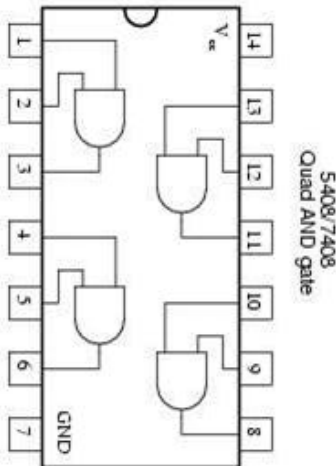
(02 Marks)

Procedure:

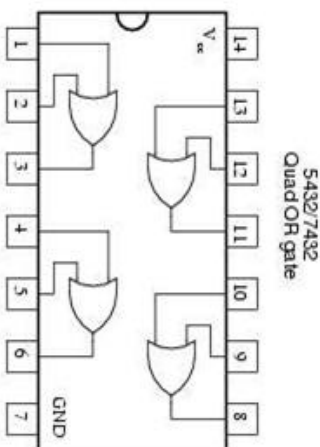
- 1) Give biasing to the ICs and all basic connections described above in the manual.
- 2) For input use switches and LEDs for outputs.
- 3) Write down the output in the tables and verify your result.

NOT Gate:**(0.5 Marks)**

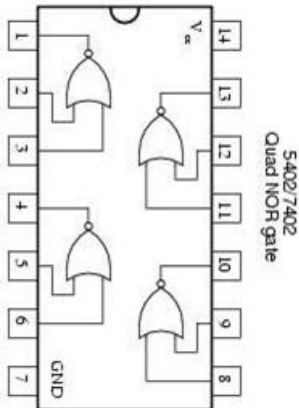
Logical NOT Gate	
A	$Y=A'$

AND Gate:**(0.5 Marks)**

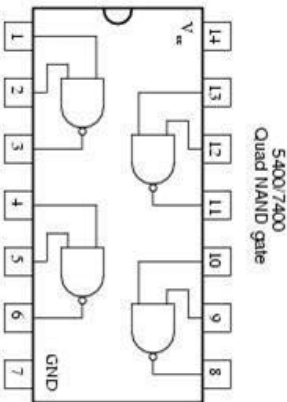
Logical AND Gate		
A	B	$Y=A.B$

OR Gate:**(0.5 Marks)**

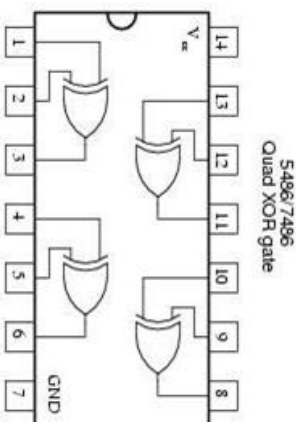
Logical OR Gate		
A	B	$Y=A+B$

NOR Gate:**(0.5 Marks)**

Logical NOR Gate		
A	B	$Y=(A+B)'$

NAND Gate:**(0.5 Marks)**

Logical NAND Gate		
A	B	$Y=(A.B)'$

XOR Gate:**(0.5 Marks)**

Logical XOR Gate		
A	B	$Y=A\oplus B$

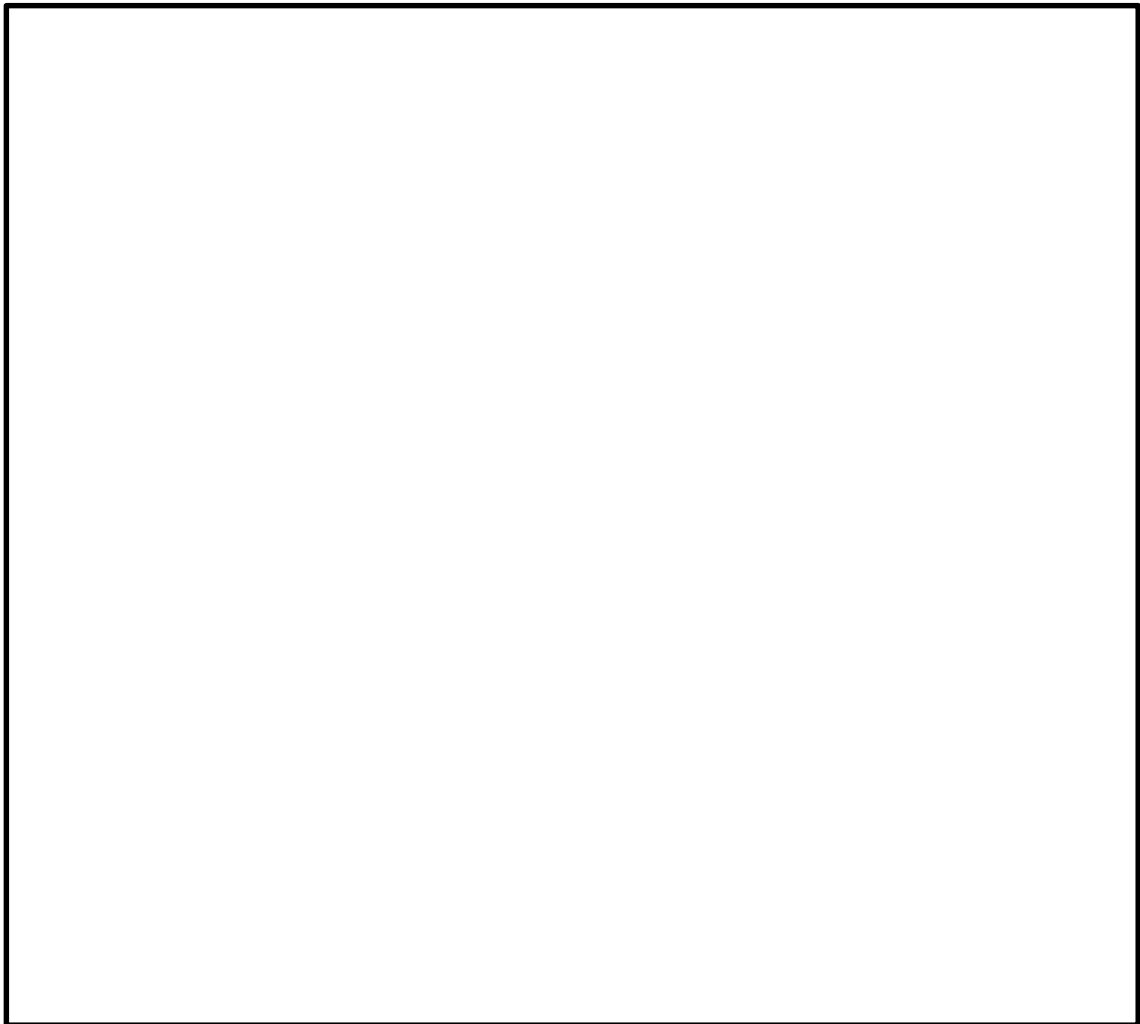
Observations:

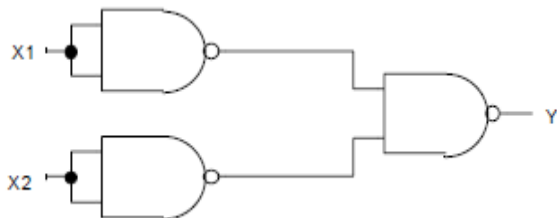
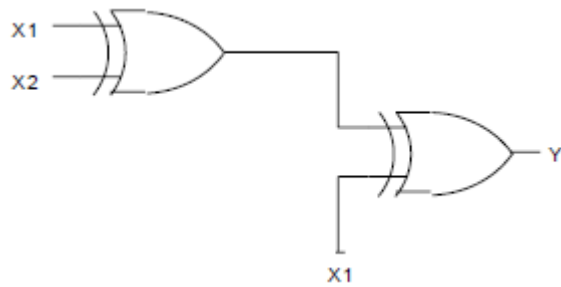
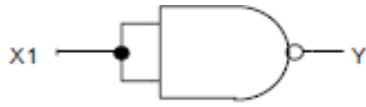
(02 Marks)

Review Questions

1. What is meant by “74LS”? (01 Mark)

2. A burglar alarm for a car has a normally low switch on each of four doors. If any door is opened, the output of that switch goes HIGH. The alarm is set off with an active-LOW output signal. What type of gate will provide this logic? Support your answer with an explanation. (01 Mark)



3. Write Truth Table for given circuits.**(01 Mark)**

BOOLEAN FUNCTION SIMPLIFICATION & IMPLEMENTATION USING BOOLEAN ALGEBRA

EXPERIMENT NO. 03

Name	Report Marks (05)	Lab Performance (05)	Viva Marks (05)	Total (15)

EXPERIMENT NO. 03

Boolean Function Simplification & Implementation using Boolean Algebra

Objectives

- Learn how to reduce the gates by function simplification
- Design of logic circuit defined by simplified function

Components

- ICs – 7404, 7432, 7408
- Wire Stripper
- Prototyping board

Theory

A function could be simplified by using properties and postulates resulting reduction in the number of gates but giving the same input as before simplification.

1. $A+0 = A$
2. $A+1 = 1$
3. $A \cdot 0 = 0$
4. $A \cdot 1 = A$
5. $A+A = A$
6. $A+A' = 1$
7. $A \cdot A = A$
8. $A \cdot A' = 0$
9. $(A')' = A$
10. $A+AB = A$
11. $A+A'B = A+B$
12. $(A+B)(A+C) = A+BC$
13. $A' \cdot B' = (A+B)'$
14. $A'+B' = (A \cdot B)'$

LAB TASK

You have to simplify two functions:

$$F_1 = A'B'C + A'BC + AB'$$

$$F_2 = xyz + xy'z + xyz$$

Follow the following steps:

- 1) Draw the truth table of the given function
- 2) Draw the circuit diagram
- 3) Simplify the function
- 4) Draw the truth table of simplified function
- 5) Compare its output with the truth table in step 1
- 6) Draw the circuit diagram of simplified function
- 7) Now patch the circuit in step 6 on breadboard and compare the outputs with the truth table in step 1 & 5

Calculations:

Function 1:

(2.5 Marks)

$$F_1 = A'B'C + A'BC + AB'$$

Truth Table

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				

Circuit Diagram:

--

Simplification:

--

Simplified Truth Table:

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				

Simplified Circuit Diagram:**Function 2:****(2.5 Marks)**

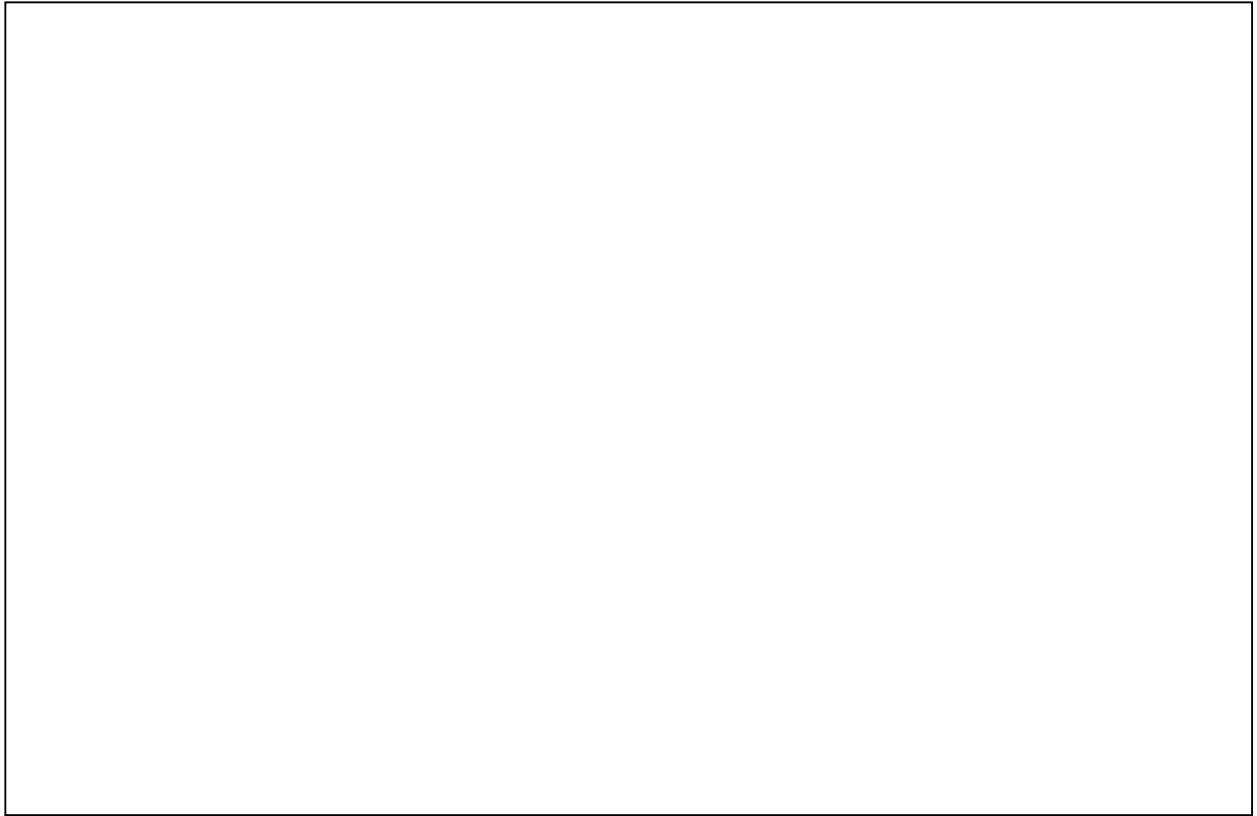
$$F_2 = xyz + xy'z + xyz$$

Truth Table:

x	y	z				
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Circuit Diagram:**Simplification:****Simplified Truth Table:**

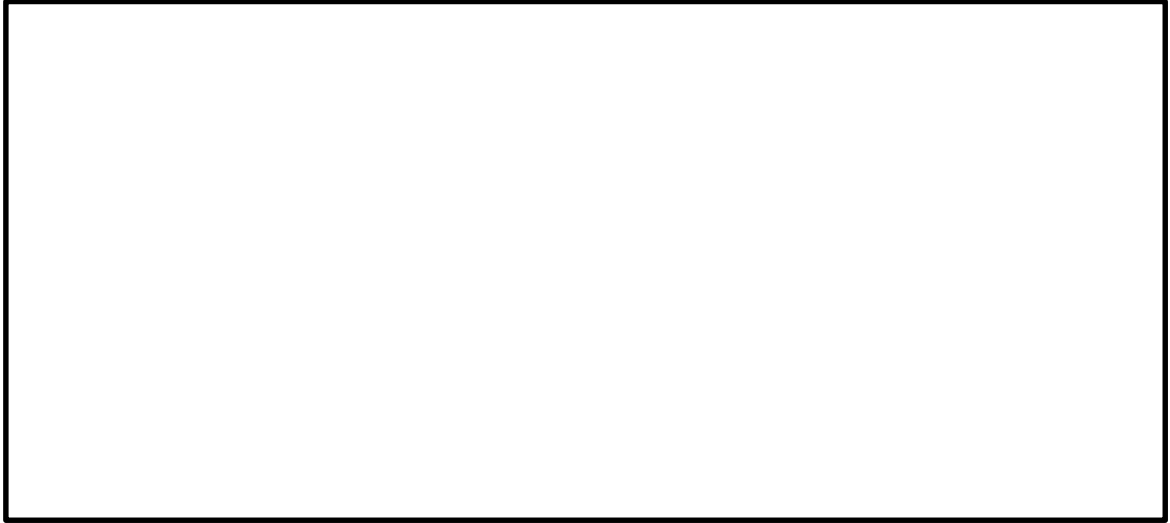
x	y	z				
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Simplified Circuit Diagram:**Observations:****(02 Marks)**

Review Questions

1. Find complement of following function and draw circuit of resultant. (01 Mark)

$$F=XY'Z+X'Y+XY'Z'$$



2. State the purpose of reducing Boolean function into minimal form. (01 Mark)

3. What is meant by duality principal? What is its usage in Boolean algebra? Explain your answer. (01 Mark)

DERIVATION OF BOOLEAN FUNCTION FROM TRUTH TABLE USING MINTERMS, SIMPLIFICATIONS & VERIFICATION

EXPERIMENT NO. 04

Name	Report Marks (06)	Lab Performance (05)	Viva Marks (04)	Total (15)

EXPERIMENT NO. 04

Derivation of Boolean Function From Truth Table Using Minterms, Simplifications & Verification

Objectives

- Learn how to write a function using minterms
- Gates reduction by function simplification

Components

- ICs – 7404, 7432, 7408
- Wire Stripper
- Prototyping board
-

Theory

Minterms provide a way to represent any Boolean function algebraically, once its truth table is specified. The function is given by the sum (OR) of those minterms corresponding to rows where the function is 1. By the minterms property, the OR will contain a term equal to 1 (making the function 1) on exactly those rows where the function is supposed to be 1.

LAB TASK

You have to write two functions (F1 & F2) using minterms and simplify them:

x	y	z	F1	F2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	0	0
1	1	1	1	0

Follow the following steps:

1. Write the Minterms of the rows where the function is 1 in the given truth table
2. Simplify the function
3. Draw the circuit diagram
4. Now patch the circuit on breadboard and compare the outputs with the truth table given

Calculations:

Function 1

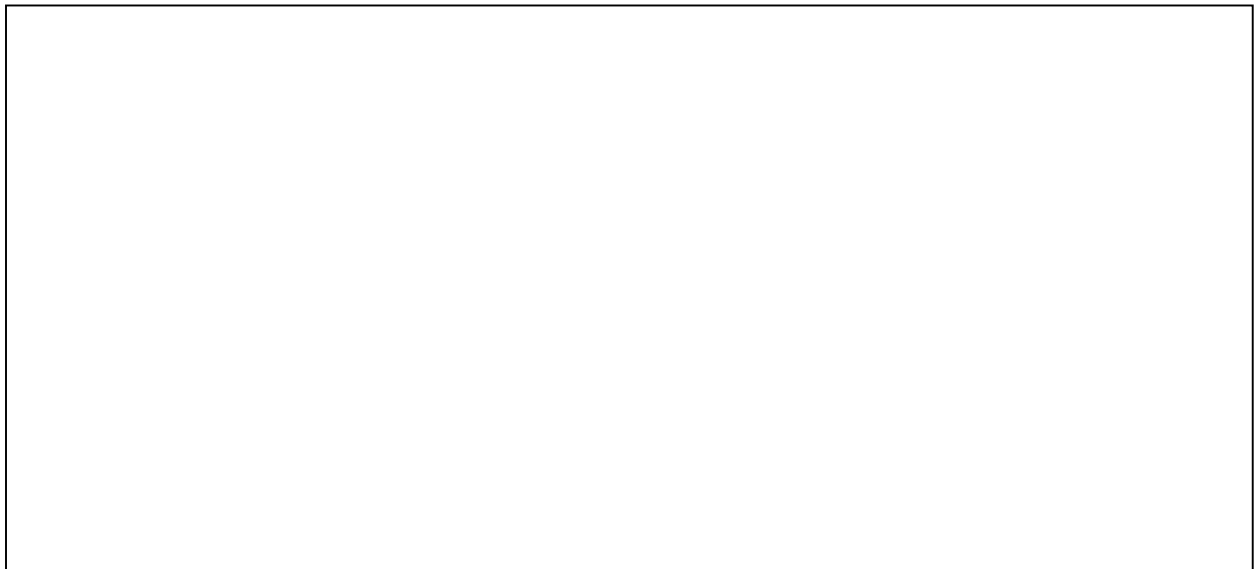
(2.5 Marks)

Function:

Simplification:



Circuit Diagram:



Function 2

(2.5 Marks)

Function:

Simplification:

A large empty rectangular box with a black border, intended for the student to show their simplification work.

Circuit Diagram:

A large empty rectangular box with a black border, intended for the student to draw their circuit diagram.

Observations:

(02 Marks)

Review Questions

- 1. What do you understand by truth table and truth function? How these are related? (0.5 Marks)**

- 2. What do you mean by “Logical Functions”? Give some real life examples that best describe them. (0.5 Marks)**

- 3. Explain the significance of truth table. (0.5 Marks)**

- 4. Consider the following situation (1.5 Marks)**
Four chairs A, B, C and D are placed in a row. Each chair may be occupied (“1”) or empty (“0”). A Boolean function “F” is 1 if and only if there are two or more adjacent chairs are occupied.

- a) Give the truth table defining this Boolean function F.
b) Express F in terms of Algebraic Function.
c) Simplify the result and show its equivalent circuit diagram

BOOLEAN FUNCTION IMPLEMENTATION USING UNIVERSAL GATE

EXPERIMENT NO. 05

Name	Report Marks (05)	Lab Performance (05)	Viva Marks (05)	Total (15)

EXPERIMENT NO. 05

Boolean Function Implementation Using Universal Gate

Objectives

- Introduction to Universal gates.
- Learn how to make gates using Universal gates.
- Implementation of a basic gate's using Universal Gates.

Theory

NAND and NOR gates are the two basic logic gates (the other being OR logic) from which any other logic gates can be built. Due to this property, NAND and NOR gates are sometimes called "universal gates".

NAND Logic

NOT

A NOT gate is made by joining the inputs of a NAND gate. Since a NAND gate is equivalent to an AND gate followed by a NOT gate, joining the inputs of a NAND gate leaves only the NOT part.

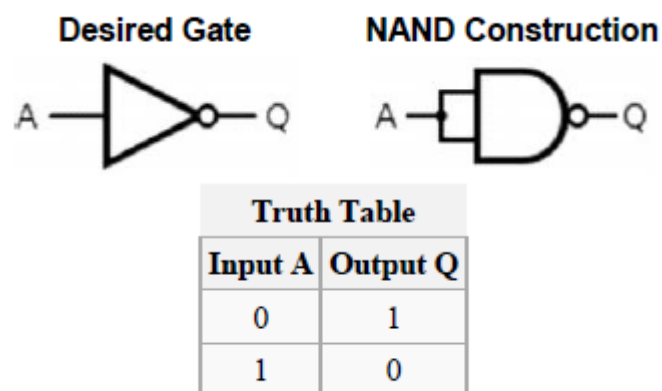


Fig 6.1: NAND to NOT

AND

An **AND** gate is made by following a NAND gate with a NOT gate as shown below. This gives a NOT NAND, i.e. AND.

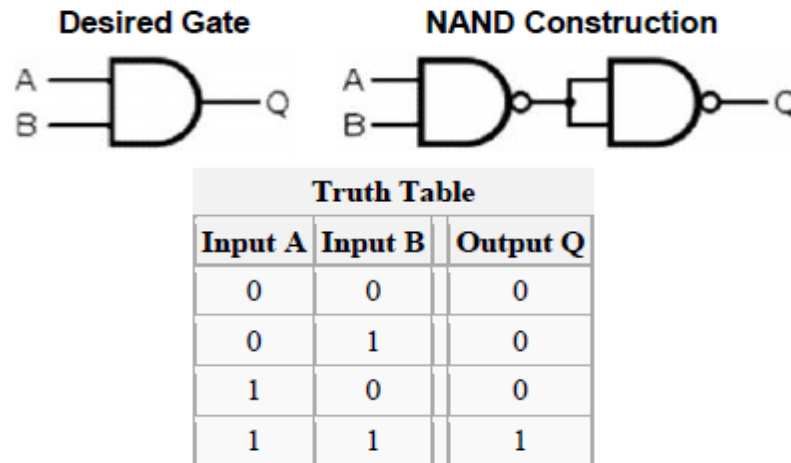


Fig 6.2: NAND to AND

OR

If the truth table for a NAND gate is examined or by applying De Morgan's Laws, it can be seen that if any of the inputs are 0, then the output will be 1. However to be an OR gate, if any input is 1, the output must also be 1. Therefore, if the inputs are inverted, any high input will trigger a high output.

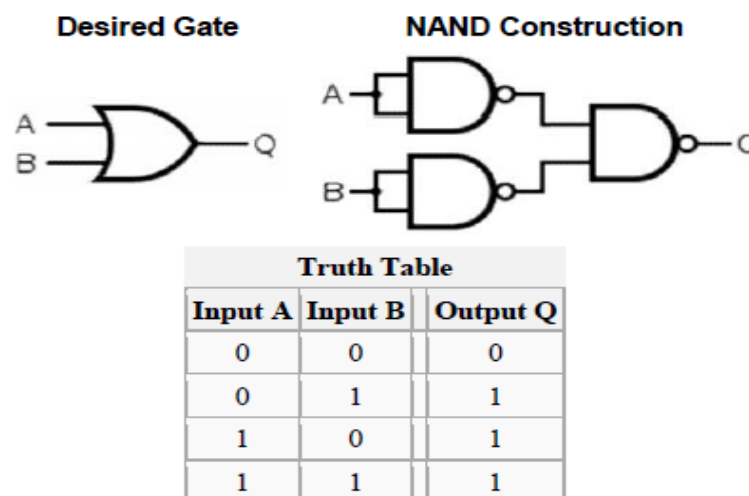


Fig 6.3: NAND to OR

NOR

A NOR gate is simply an OR gate with an inverted output:

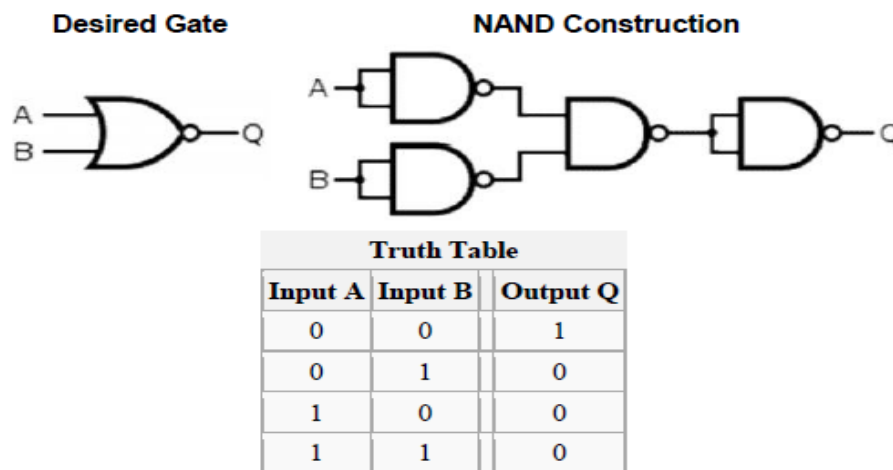


Fig 6.4: NAND to NOR

XOR

An XOR gate is constructed similarly to an OR gate, except with an additional NAND gate inserted such that if both inputs are high, the inputs to the final NAND gate will also be high, and the output will be low. This effectively represents the formula: "NAND(A NAND (A NAND B)) NAND (B NAND (A NAND B))".

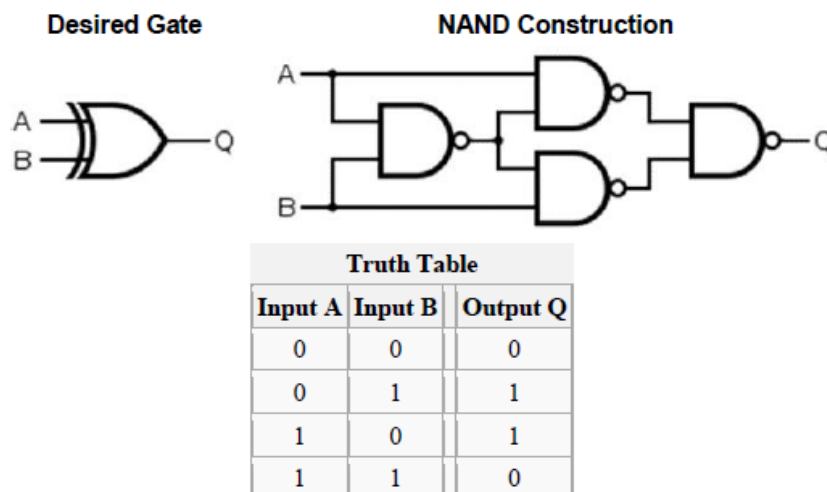


Fig 6.5: NAND to XOR

NOR Logic

A NOR gate is logically an inverted OR gate. By itself has the following truth table:

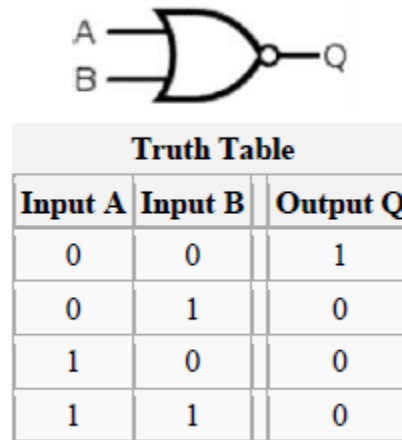


Fig 6.6: A simple NOR gate

NOT

This is made by joining the inputs of a NOR gate. As a NOR gate is equivalent to an OR gate leading to NOT gate, this automatically sees to the "OR" part of the NOR gate, eliminating it from consideration and leaving only the NOT part.

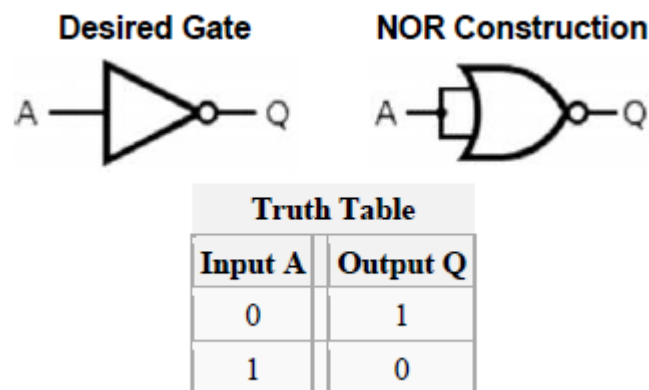
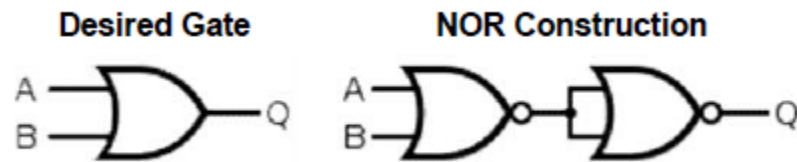


Fig 6.7: NOR to NOT

OR

The OR gate is simply a NOR gate followed by a NOT gate.

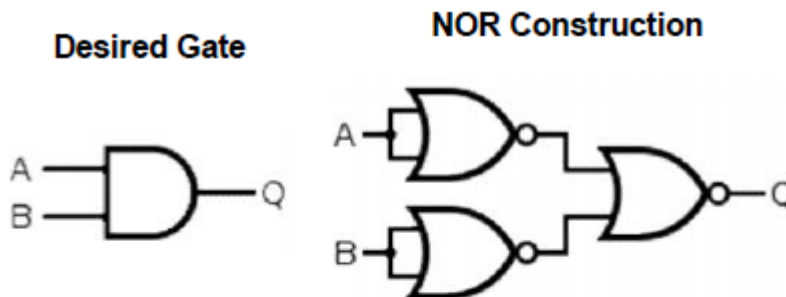


Truth Table		
Input A	Input B	Output Q
0	0	0
0	1	1
1	0	1
1	1	1

Fig 6.8: NOR to OR

AND

An AND gate gives a 1 output when both inputs are 1; a NOR gate gives a 1 output only when both inputs are 0. Therefore, an AND gate is made by inverting the inputs to a NOR gate.



Truth Table		
Input A	Input B	Output Q
0	0	0
0	1	0
1	0	0
1	1	1

Fig 6.9: NOR to AND

NAND

A NAND gate is made using an AND gate in series with a NOT gate:

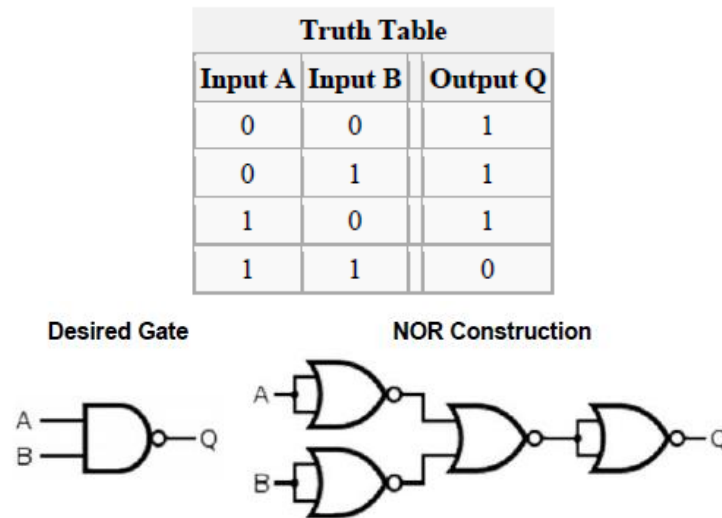


Fig 6.10: NOR to NAND

XOR

An XOR gate is made by connecting the output of 3 NOR gates (connected as an AND gate) and the output of a NOR gate to the respective inputs of a NOR gate. This expresses the logical formula $(A \text{ AND } B) \text{ NOR } (A \text{ NOR } B)$. This construction entails propagation delay three times that of a single NOR gate.

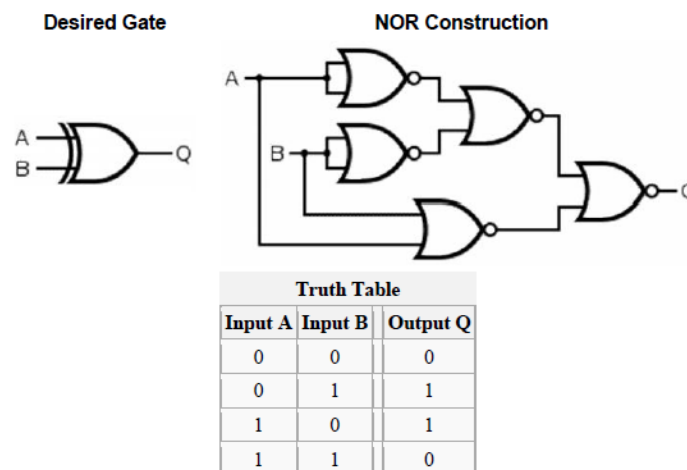


Fig 6.11: NOR to XNOR

Lab Tasks

- 1) Implement XNOR gate using NAND gates & Write truth table and draw circuit diagram. (2 Marks)
- 2) Implement XNOR gate using NOR gates & Write truth table and draw circuit diagram. (2 Marks)
- 3) Implement XOR gate using NAND and NOR gates & Write truth table and draw circuit diagram. (2 Marks)

Components

- ICs – 7400 and 7402
- Wire Stripper
- Prototyping board with power and ground connections

Procedure

- Give biasing to the ICs and all basic connections described above in the manual for given tasks.
- Give all possible combinations of input and note down the outputs.
- For input use switches and LEDs for outputs.

Calculations:

1. Fill in the truth table for the XNOR function using NAND Gate

Input A	Input B	Output F

Write Boolean function for XNOR gate: **F** = -----

Draw NAND logic diagram for the XNOR gate:

2. Fill in the truth table for the XNOR function using NOR Gate

Input A	Input B	Output F

Write Boolean function for XNOR gate: **F** = -----

Draw NOR logic diagram for the XNOR gate:

3. Fill in the truth tables for the XOR function using NOR & NAND Gates**NOR Logic**

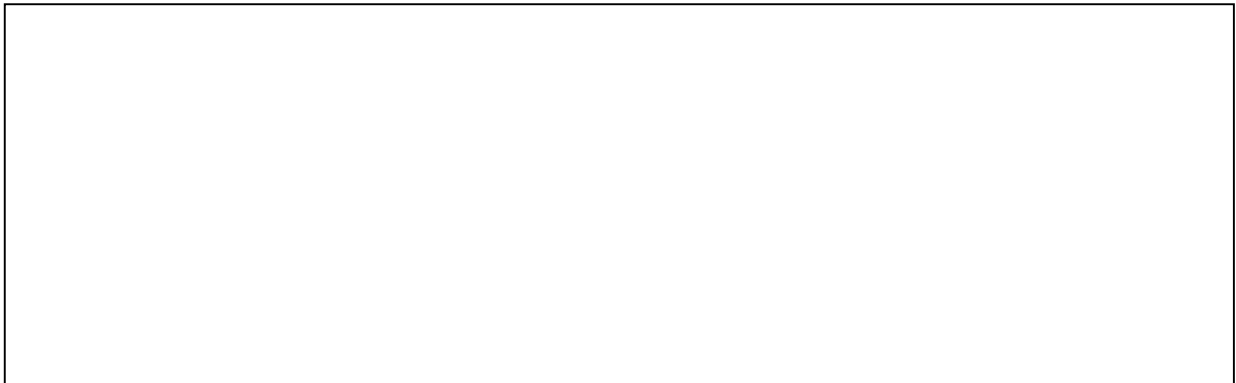
Input A	Input B	Output F

NAND Logic

Input A	Input B	Output F

- Write Boolean function for XNOR gate NOR Logic : $F = \text{-----}$
- Write Boolean function for XNOR gate NAND Logic: $F = \text{-----}$

Draw NOR logic diagram for the XOR gate:



Draw NAND logic diagram for the XOR gate:



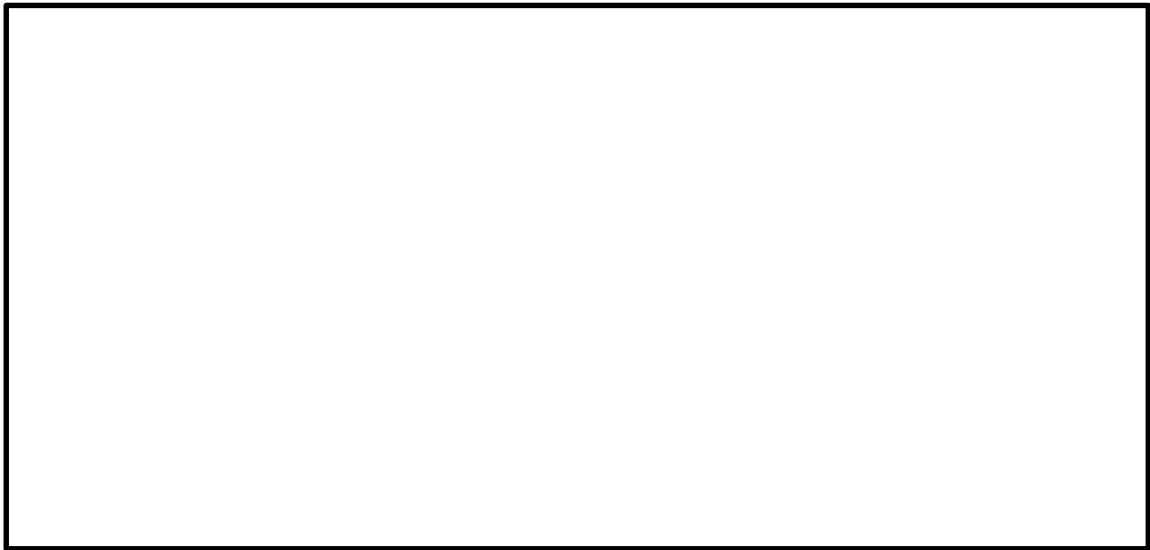
Observations:

(02 Marks)


Review Questions

- 1. Why are the NAND and NOR gates more popular? (01 Mark)**

- 2. Draw logical circuit for following using NAND only $XY+XY'Z+XYZ$ (01 Mark)**



- 3. Draw the logical function using NOR only $(X+Y+Z).(X+Y'+Z')$ (01 Mark)**



HARDWARE IMPLEMENTATION AND VERIFICATION OF ENCODER AND DECODER

EXPERIMENT NO. 06

Name	Report Marks (5)	Lab Performance (5)	Viva Marks (5)	Total (15)

EXPERIMENT NO. 06

Hardware Implementation and Verification of Encoder and Decoder

Objectives

- Introduction to Encoders and Decoders.
- Implementation of Encoders and Decoders.

Theory

Decoder:

The process of taking some type of code and determining what it represents in terms of a recognizable number or character is called decoding. A decoder is a combinational logic circuit that performs the decoding function, and produce an output that indicates the (meaning) of the input code.

The decoder is an important part of the system which selects the cells to be read from and write into. This particular circuit is called a decoder matrix.

Decoder is a circuit that convert binary information from n-input lines to max of 2^n output lines e.g. if we have 2 inputs i.e. x, y then there will be 4 output of a Decoder and size of Decoder will be 2×4 .

$$N \rightarrow 2^n$$

n= No. of input lines.

2^n = No. of outputs of a Decoder.

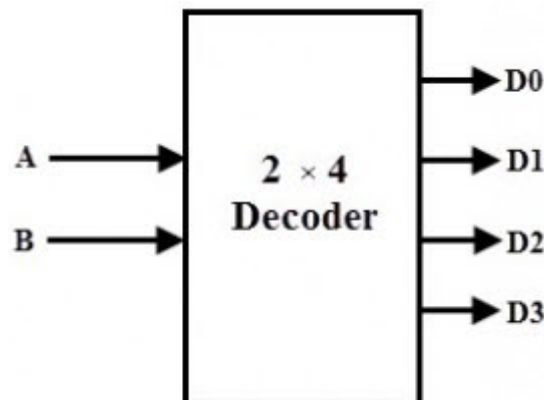


Fig 10.1: 2 x 4 Decoder

Encoder:

Encoders work in exactly the opposite way as decoders, taking 2^n inputs, and having n outputs. When a bit comes in on an input wire, the encoder outputs the physical address of that wire. It takes 2^n inputs and gives out n outputs; the enable pin should be kept 1 for enabling the circuit.

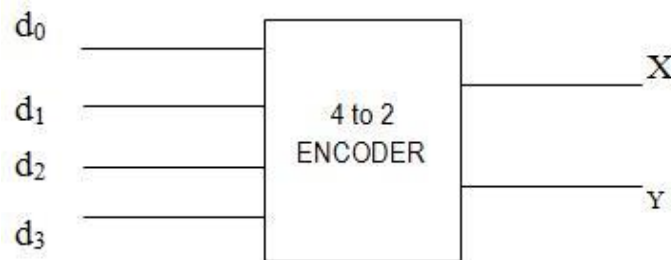
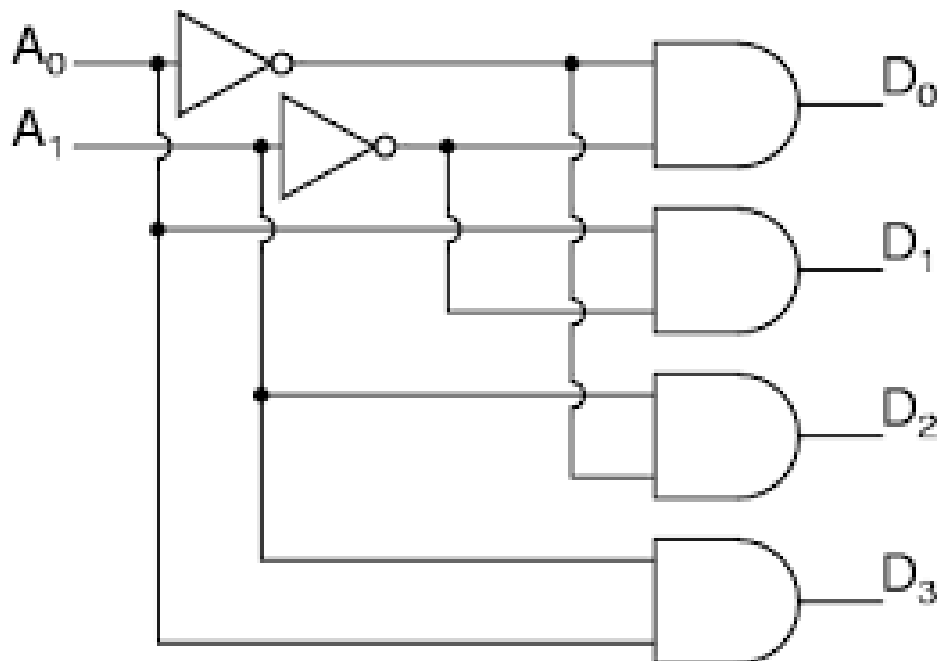


Fig 10.2: 4 x 2 Encoder

LAB Task

- To setup Decoder & Encoder fill up the Truth Tables and write expressions **(05 Marks)**

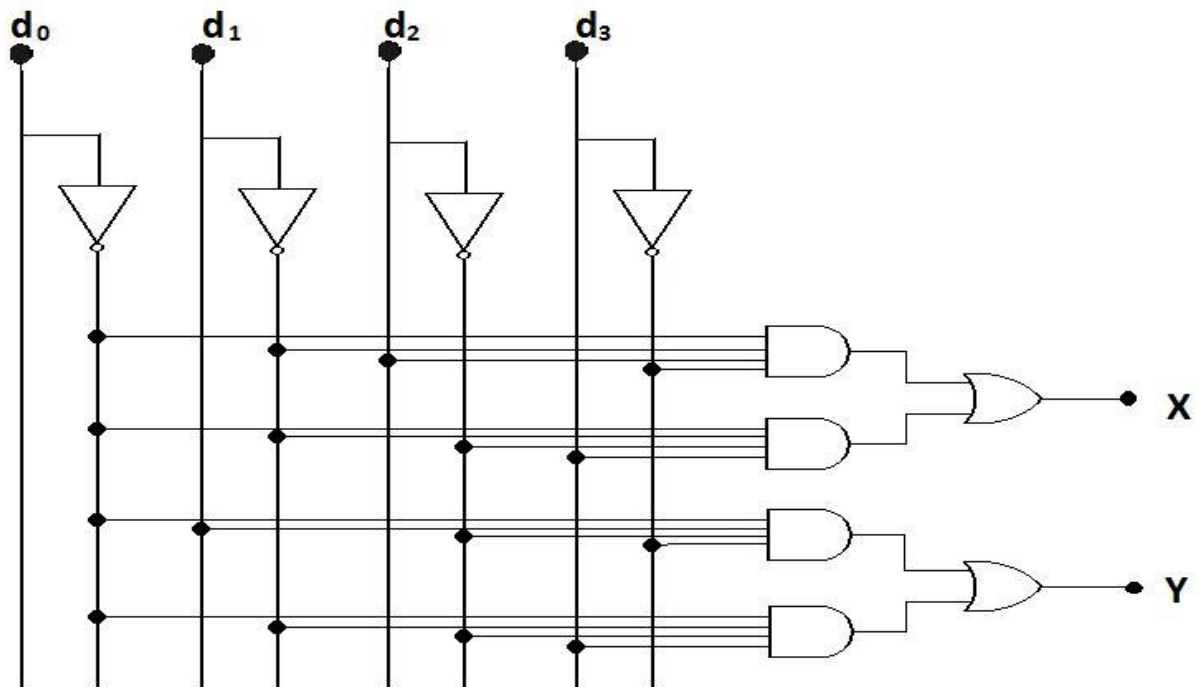
Decoder:

- **Hardware Connections**

(01 Mark)**(01 Mark)**

Input		Output			
A ₀	A ₁	D ₀	D ₁	D ₂	D ₃

Write the Boolean Function for 2 x 4 Decoder**(0.5 Marks)**

Encoder:

- **Hardware Connections**

(01 Mark)**(01 Mark)**

Inputs				Outputs	
D0	D1	D2	D3	X	Y

Write the Boolean Function for 4 x 2 Encoder**(0.5 Marks)**

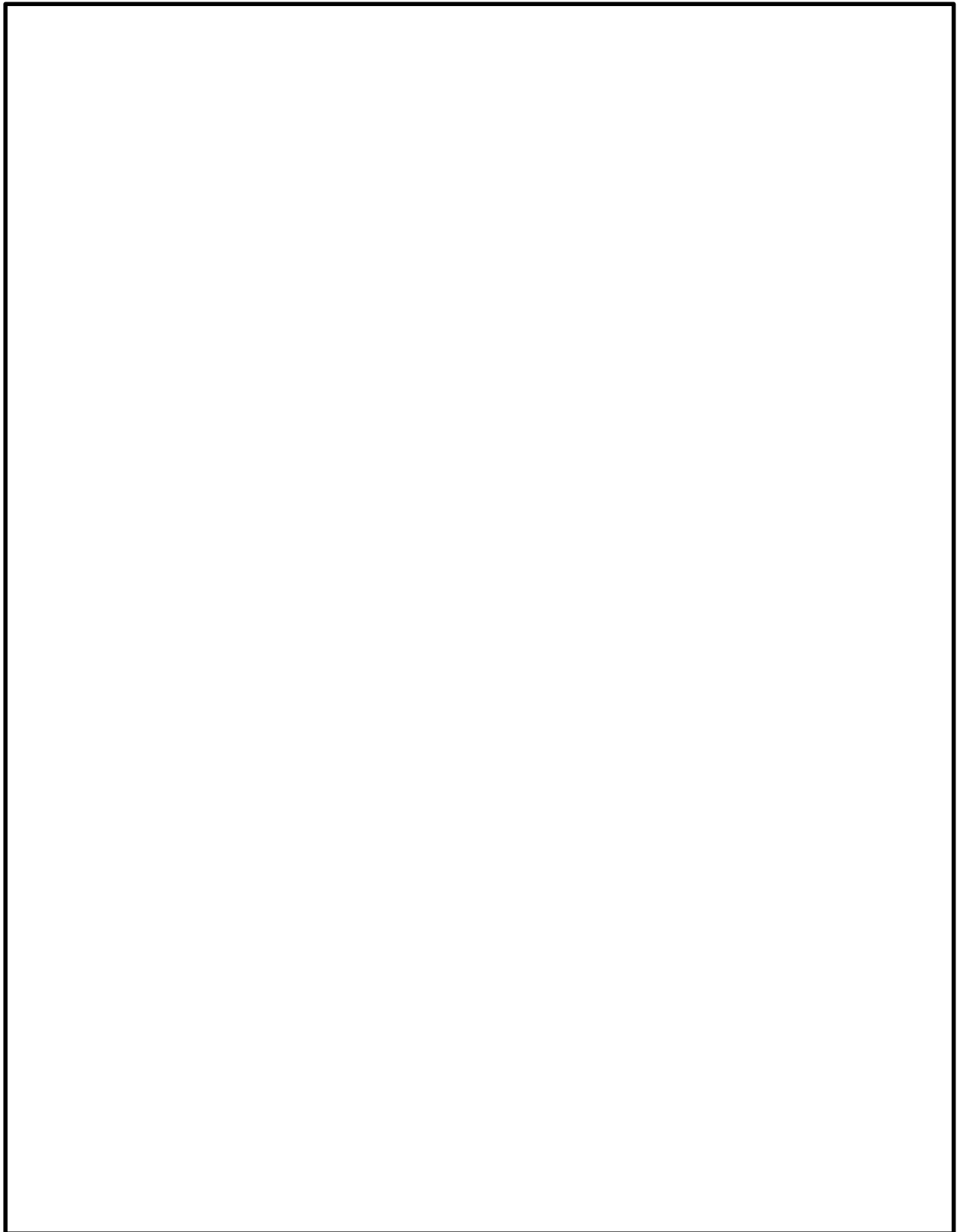
Observations:

(02 Marks)

Review Questions

- 1 **Design 3×8 decoder from 2×4 decoder.** **(01 Mark)**

- 2 **Design 4×16 decoder from 3×8 decoder.** **(01 Mark)**



MULTIPLEXER & DEMULTIPLEXER

EXPERIMENT NO. 07

Name	Report Marks (5)	Lab Performance (5)	Viva Marks (5)	Total (15)

EXPERIMENT NO. 07

Multiplexer and De-Multiplexer

Objectives

- Introduction to Multiplexer and DeMultiplexer.
- Learn the uses of Multiplexer and DeMultiplexer.
- Implementation of Mux and DeMux and their verification using truth table.

Components

- ICs – 7408, 7432 and 7404
- Wire Stripper
- Prototyping board with power and ground connections

Theory

Multiplexer

Multiplexer, simply called MUX, is a data selector and is capable of “selecting” one of many input lines (usually 2^n) and display its input status on the only output line available.

A MUX has

- 1) Select lines
- 2) Data input lines
- 3) Output line.

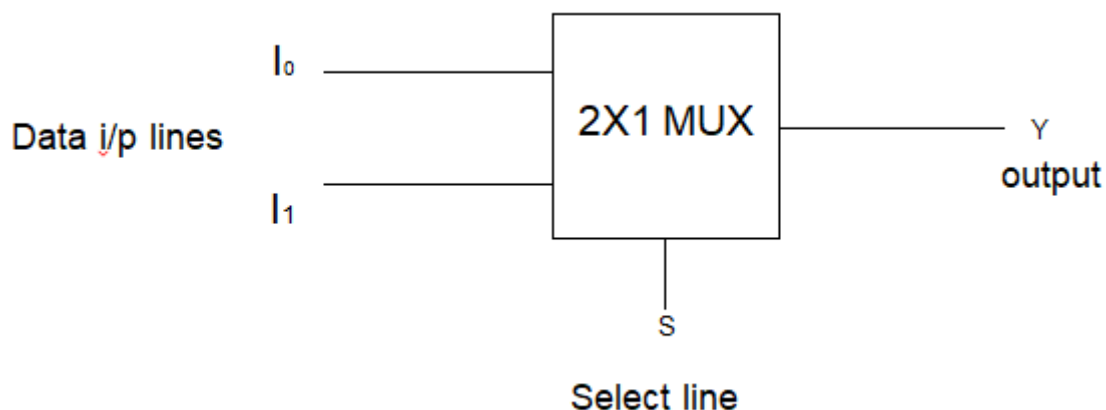


Fig 12: 2 X 1 MUX

THE FUNCTION TABLE OF 2X1 MUX IS

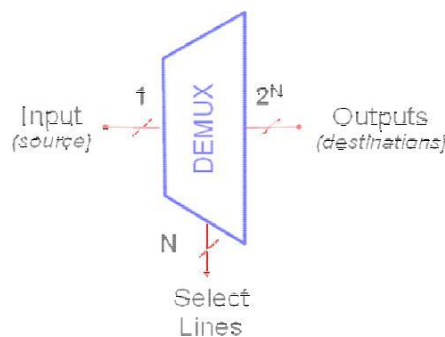
Select line	Output
S	Y
0	I ₀
1	I ₁

THE BOOLEAN FUNCTION FOR 2X1 MUX IS

$$Y = I_1 s + I_0 S'$$

DeMultiplexer

A DEMUX is a digital switch with a single input (source) and a multiple outputs (destinations).the select lines determine which output the input is connected to.

**Fig 12: DeMUX****THE FUNCTION TABLE OF 1X2 De MUX IS**

Select line	Output
S	Y
0	D ₀
1	D ₁

THE BOOLEAN FUNCTION FOR 2X1 MUX IS

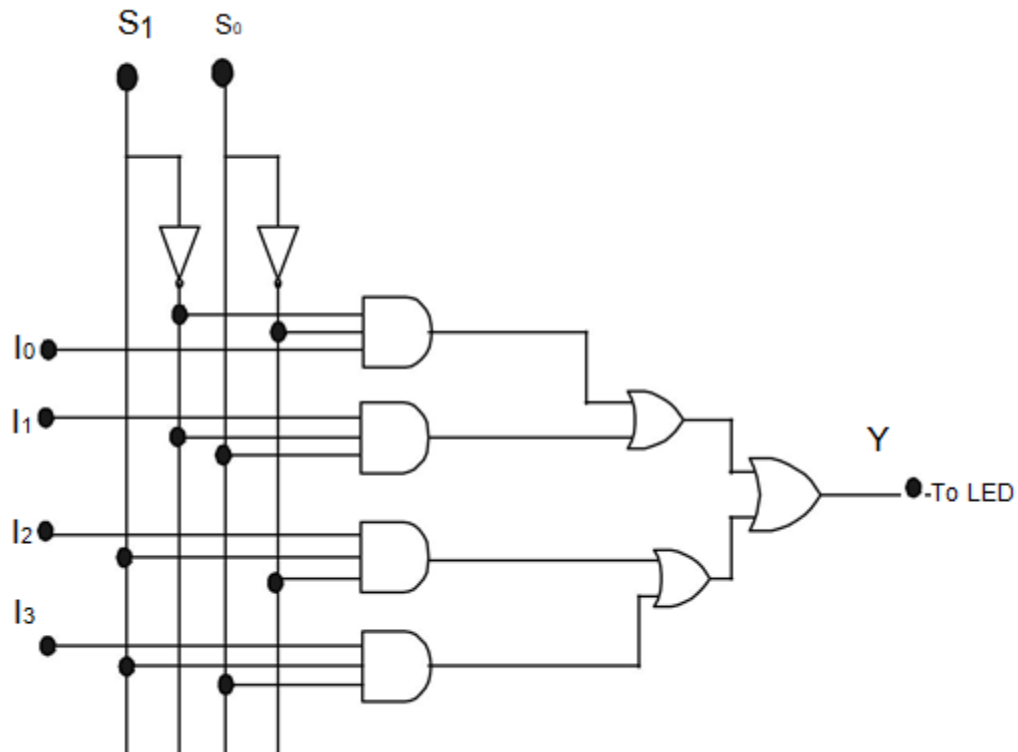
$$D_0 = I_0 S'$$

$$D_1 = I_1 S$$

LAB Task

- Implement 4 x 1 Multiplexer (2.5 Marks)
- Implement 1 x 4 De-Multiplexer (2.5 Marks)

4 x 1 Multiplexer:



Fill up the Function Table of 4X1 MUX

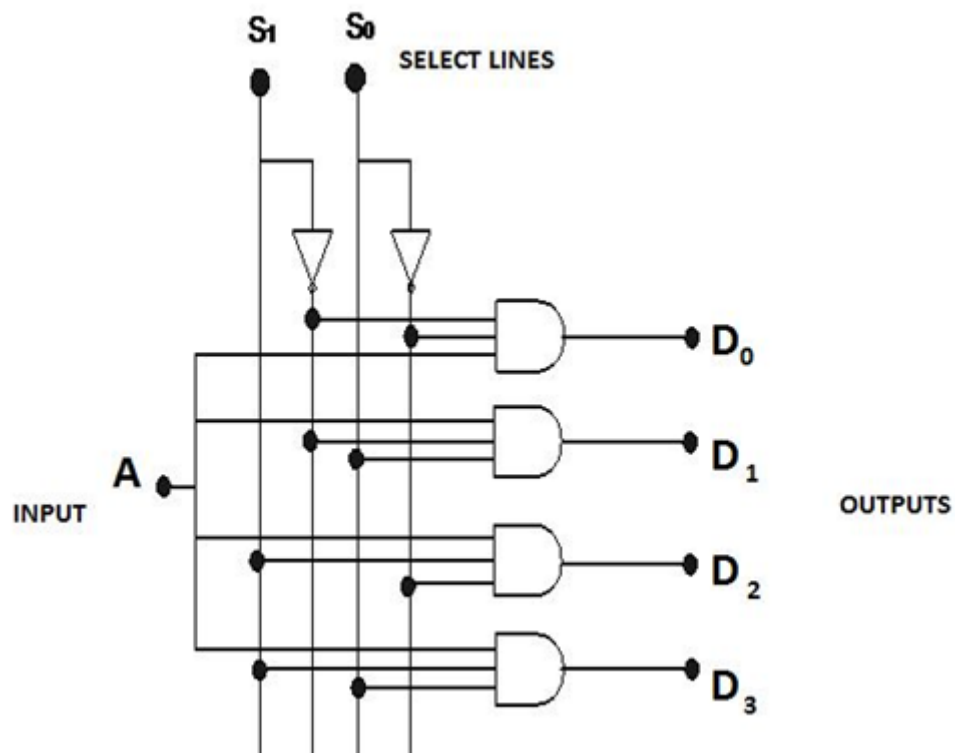
(01 Mark)

Select line		Output
S ₀	S ₁	Y

Write the Boolean Function for 4 X 1 Multiplexer:

(1.5Marks)

1 X 4 DeMultiplexer:



Fill up the Function Table of 1 X 4 DeMUX:

(01 Mark)

Select line		Output
S ₀	S ₁	D

Write the Boolean Function for 1 X 4 DeMultiplexer:

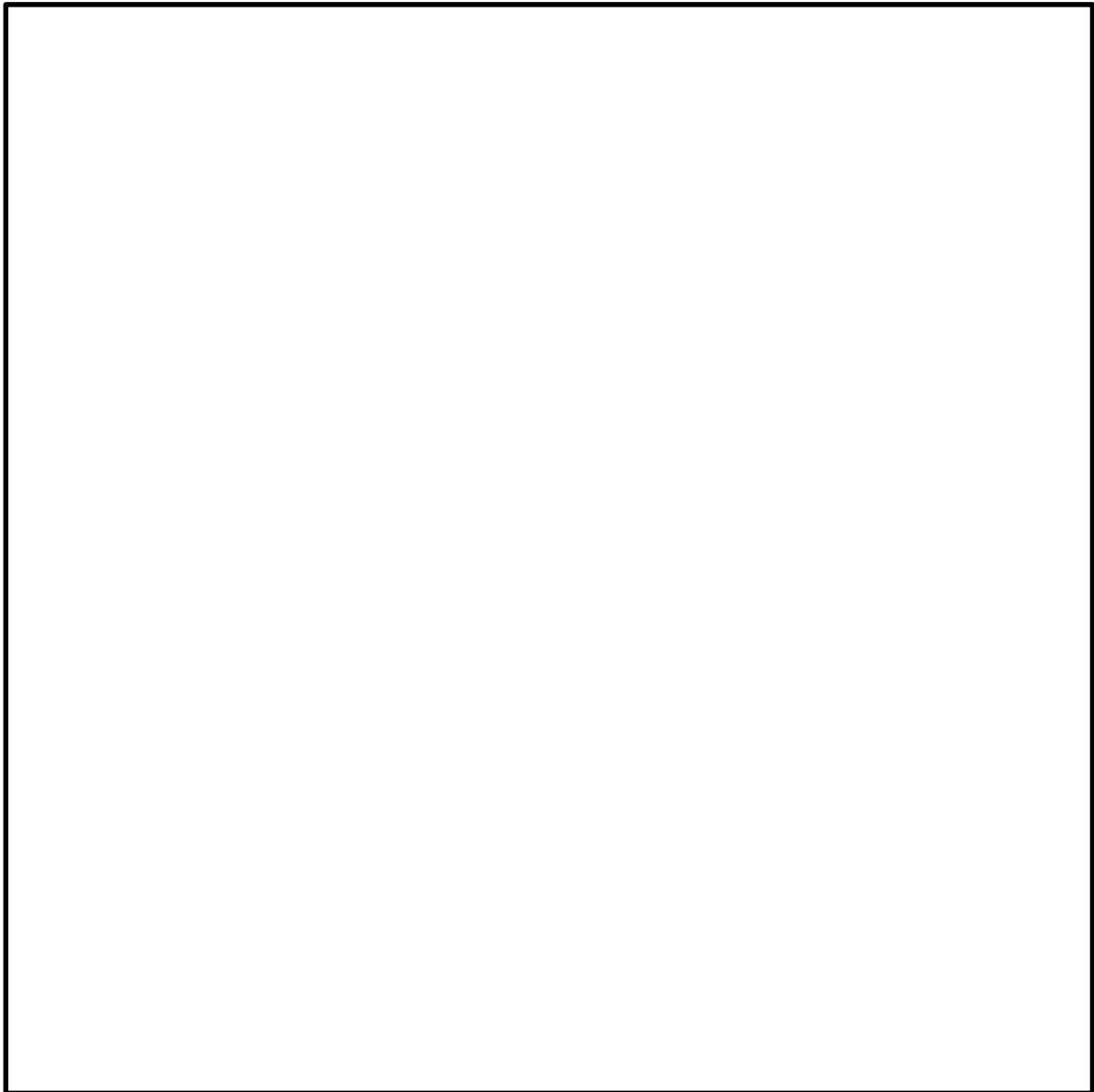
(1.5Marks)

Observations:

(02 Marks)

Review Questions

- 1) Implement the 2-to-1 multiplexer and 1-to-2 DeMultiplexer. (01 Mark)



- 2) What are the applications of multiplexer? (01 Mark)

ADDER & SUBTRACTOR

EXPERIMENT NO. 08

Name	Report Marks (5)	Lab Performance (5)	Viva Marks (5)	Total (15)

EXPERIMENT NO. 08

Adder & Subtractor

Objectives

- Learn how to implement Half and Full Adders and Subtractors

Theory

Adders

Half adder

The simplest binary adder is called a half adder. Half adder has two input bits and two output bits. The input variable designates the augends and the addend bits. One output is sum and other is carry. They are represented by S & C. It is necessary to have two output variables because the result may consist of two binary digits. The carry output is 0 unless both inputs are 1. The 'S' output represent the least significant bit of the sum.

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

$$C = A.B$$

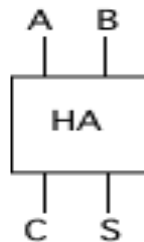


Fig 9.1: Half Adder

A half adder has no provision to add a carry from lower order bits when binary numbers are added

Full adder

A full adder is a combinational circuit that forms the arithmetic sum of three input bits. It consists of three inputs and two outputs. Two of the input variables, represent the two significant bits to be added. The third input, represents the carry from the previous lower significant position. Two outputs are necessary because the arithmetic sum of three binary digits ranges in value from 0 to 3 and binary 2 or 3 requires two bits. The two outputs are designated as S and C. The binary variable S gives the value of the least significant bit of the sum. The binary variable C gives the output carry. With three inputs, eight input combinations are possible. When all the inputs are 0, then output is 0.

The S is equal to 1 when only one input is equal to 1 or all the three inputs are equal to 1. The C output has a carry of 1 if two or three inputs are equal to 1.

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

$$C = AB + AC + BC$$

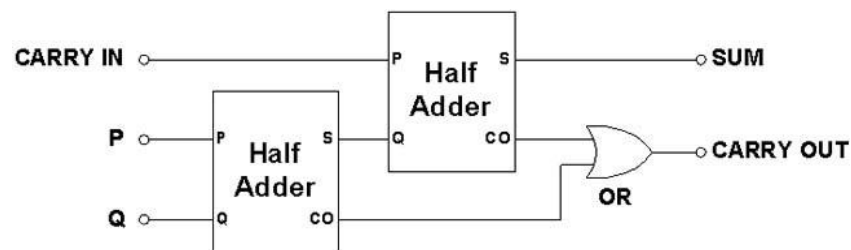


Fig 9.2: Full adder with the help of two half adders

Subtractors

Logical Subtractors

The arithmetic operation, subtraction of two binary digits has four possible elementary operations, namely,

$$0 - 0 = 0$$

$$0 - 1 = 1 \text{ with 1 borrow}$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

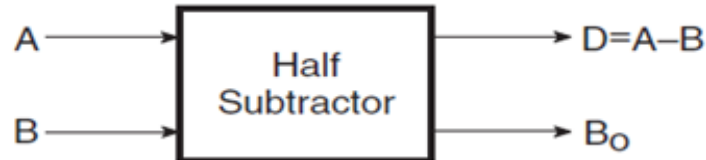
In all operations, each subtrahend bit is subtracted from the minuend bit. In case of the second operation the minuend bit is smaller than the subtrahend bit, hence 1 is borrowed.

Half Subtractor:

A combinational circuit which performs the subtraction of two bits is called half subtractor. The input variables designate the minuend and the subtrahend bit, whereas the output variables produce the difference and borrow bits.

$$D = X \oplus Y$$

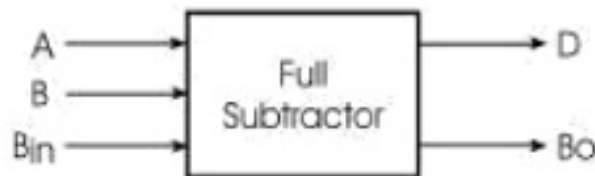
$$B_o = X'Y$$

**Fig9.6: Half Subtractor****Full Subtractor:**

A combinational circuit which performs the subtraction of three input bits is called full subtractor. The three input bits include two significant bits and a previous borrow bit. A full subtractor circuit can be implemented with two half sub-tractors and one OR gate.

$$D = (X \oplus Y) \oplus B_{in}$$

$$B_o = X.Y + (X \oplus Y) B_{in}$$

**Fig9.7: Full Subtractor**

Lab Task

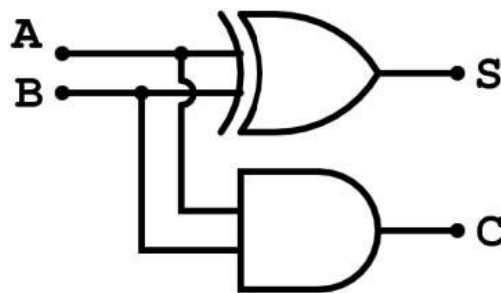
- To setup Half Adders & Full Adders fill up the Truth Table and write expressions. (2.5Marks)
- To setup Half Subtractor & Full Subtractor fill up the Truth Table and write expressions. (2.5Marks)

Components

- ICs 7486, 7432, 7408, 7404
- Wire Stripper
- Prototyping board with power and ground connections

Calculations for Half Adder & Full Adder:

- Half adder**



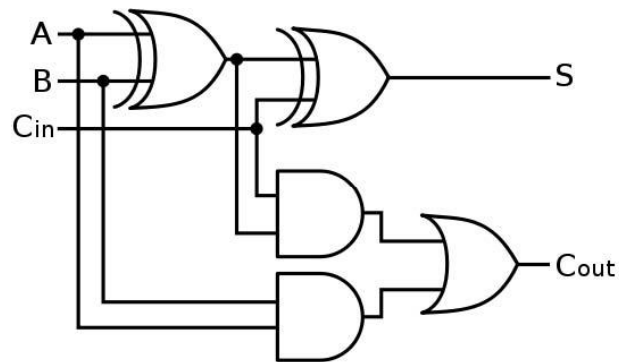
Input A	Input B	Sum (S)	Carry (C)

Write sum of products expression for SUM and CARRY:

(01 Mark)

SUM = -----

CARRY = -----

Full Adder:

(0.5 Marks)

Input A	Input B	Cin	SUM	Cout

Write and simplify sum of products expression for Sum:

(0.5 Marks)

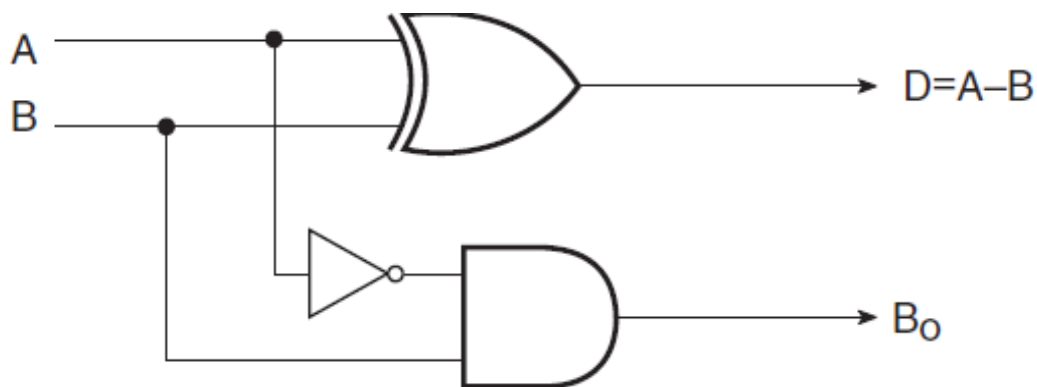
Write and simplify sum of products expression for Carry:

(0.5 Marks)

Calculations for Half Subtractor & Full Subtractor:

Half Subtractor:

A combinational circuit which performs the subtraction of two bits is called half subtractor. The input variables designate the minuend and the subtrahend bit, whereas the output variables produce the difference and borrow bits.



Input A	Input B	Difference(D)	Borrow (B)

Write sum of
expression for Difference and Borrow:

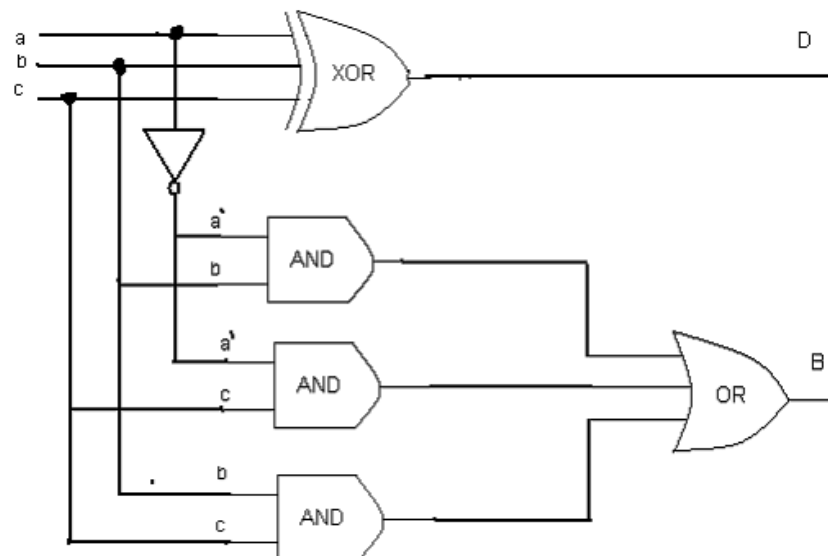
products
(01 Mark)

Difference = -----

Borrow = -----

Full Subtractor:

A combinational circuit which performs the subtraction of three input bits is called full subtractor. The three input bits include two significant bits and a previous borrow bit. A full subtractor circuit can be implemented with two half sub-tractors and one OR gate.



(0.5 Marks)

Input A	Input B	Input C	DIFFERENCE	BORROW

Write and simplify sum of products expression for Difference using K-MAP: (0.5 Marks)

--

Write and simplify sum of products expression for Borrow using K-MAP: (0.5 Marks)

--

Observations:

(02 Marks)

Review Questions

- 1. What is the main difference between the equations of adder and subtractor? Explain why is it so? (01 Mark)**

- 2. Write some applications of half and full Subtractor? (01 Mark)**

- 3. Write some applications of half and full Adder? (01 Mark)**

BINARY COMPARATOR

EXPERIMENT NO. 09

Name	Report Marks (02)	Lab Performance (08)	Viva Marks (5)	Total (15)

EXPERIMENT NO. 09**Binary Comparator****Objectives:**

- To learn and understand how to design a multiple output combinational circuit.
- To learn and understand the working of 2-bit binary comparator.
- To learn and understand the working and usage of Exclusive-OR and Exclusive-NOR gates.

COMPONENTS: ICs 74LS08, 74LS32, 74LS04, 74LS86, 74LS02

THEORY:

Binary comparator is a combinational circuit that compares magnitude of two binary data signals A & B and generates the results of comparison in the form of three output signals $A > B$, $A = B$, $A < B$. Binary comparator is a multiple input and multiple output combinational circuit. When a combinational circuit has two or more than two outputs then each output is expressed separately as a function of all inputs. Separate K-map is made for each output.

One-bit comparator:

One-bit comparator compares magnitude of two numbers A and B, 1 bit each, and generates the comparison result. The result consists of three outputs let us say L, E, G, so that

$$L = 1 \text{ if } A < B$$

$$E = 1 \text{ if } A = B$$

$$G = 1 \text{ if } A > B$$

Truth Table:

Inputs		Outputs		
A	B	L	E	G
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

K-Maps for Outputs:

		B	
		0	1
A	0		1
	1		

K-Map for Output L

		B	
		0	1
A	0	1	
	1		1

K-Map for Output E

		B	
		0	1
A	0		
	1	1	

K-Map for Output G

Boolean Expressions of Outputs:

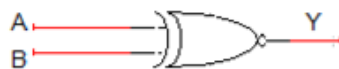
L: $\bar{A}B$

E: $AB + \bar{A}\bar{B}$

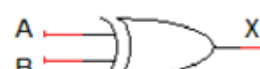
G: $A\bar{B}$

Exclusive-OR & Exclusive-NOR gates:

The figure given below shows the symbol of Exclusive-OR (XOR) and Exclusive-NOR (XNOR) gates.

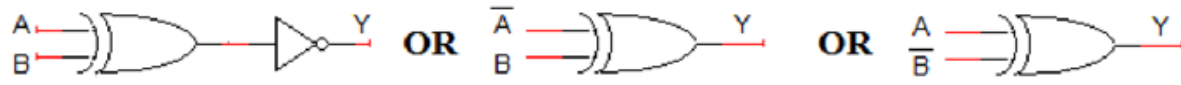


XNOR gate

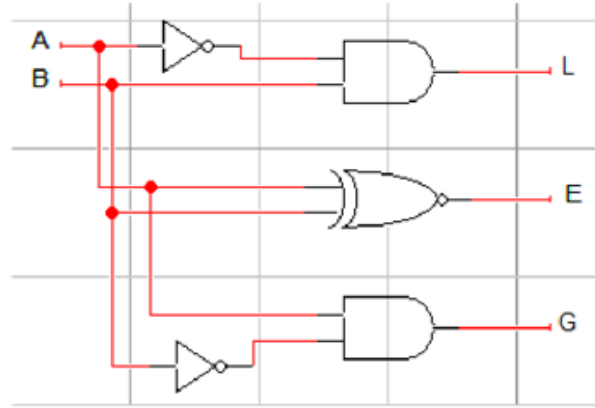


XOR gate

Boolean expression of XNOR gate is $AB + \bar{A}\bar{B}$ and Boolean expression of XOR is $\bar{A}B + A\bar{B}$.
Boolean expression of XNOR gate can be implemented using XOR gate as shown in figure below:



Circuit Diagram for one-bit comparator:



In this experiment 74LS86 IC will be used for implementation of XOR gate function. 74LS86 IC contains four 2-input XOR gates. The function table and connection diagram for this IC are shown below:

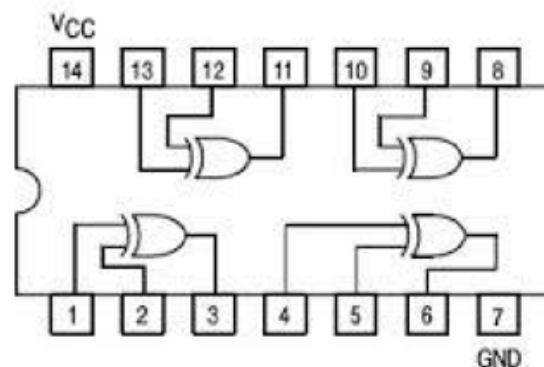
In this experiment 74LS86 IC will be used for implementation of XOR gate function. 74LS86 IC contains four 2-input XOR gates. The function table and connection diagram for this IC are shown below:

Function Table:

Inputs		Output
A	B	Y
L	L	L
L	H	H
H	L	H
H	H	L

H= Logic High, L= Logic Low

Connection Diagram:



Lab Task**(08 Marks)**

Design a combinational circuit that compares two 2-bit numbers and generates the comparison result. The result consists of three outputs let us say L, E, G, so that

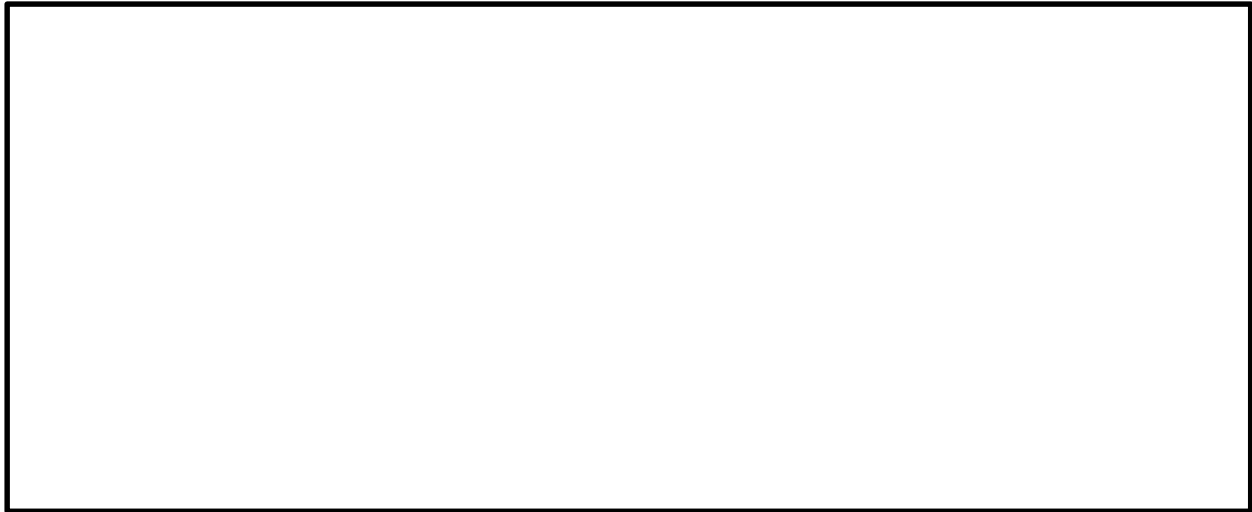
$$L = 1 \text{ if } A < B$$

$$E = 1 \text{ if } A = B$$

$$G = 1 \text{ if } A > B$$

- Write Truth table

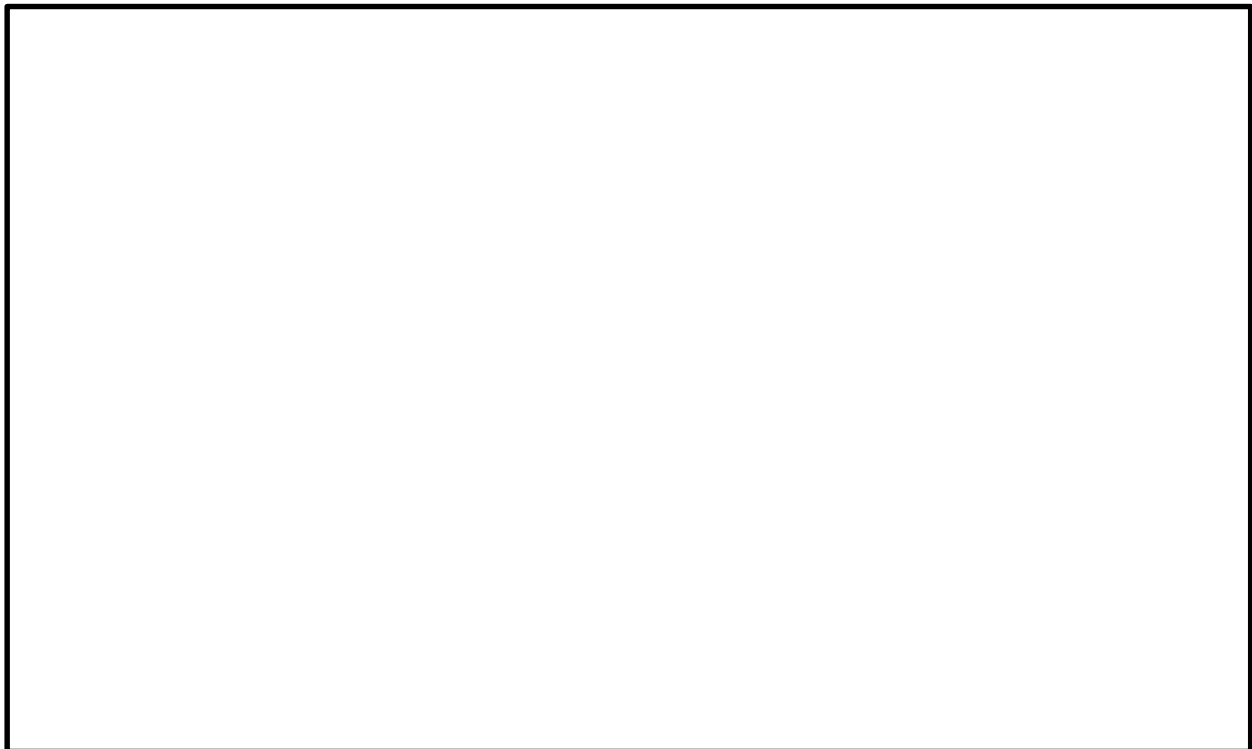
(02 Marks)**K-Maps for output:****(02 Marks)**



- Boolean expression of outputs (02 Marks)



- Circuit Diagram (02 Marks)



Observations:

(02 Marks)

INTRODUCTION TO LOGISIM SOFTWARE

EXPERIMENT NO. 10

Name	Report Marks (5)	Lab Performance (5)	Viva Marks (5)	Total (15)

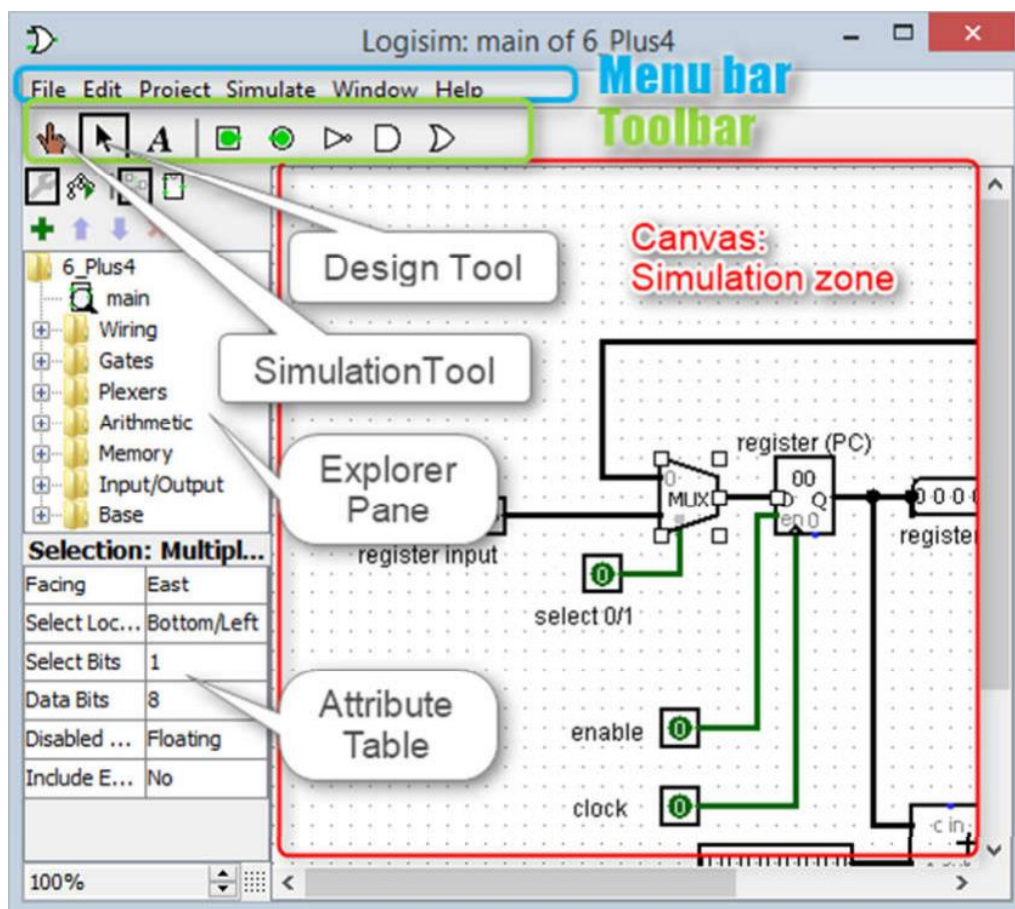
EXPERIMENT NO. 10**Introduction to Logisim Software & Simulation Of
Combinational Logic Circuit****Objectives:**

- Introduction to Logisim, a digital logic simulator
- Implementation Combinational

Theory

Logisim is a program that lets you build virtual logic circuits using gates, clock and many other things from digital logic design.

When you start Logisim, you'll see a window similar to the following. Since you'll be using a different system, some of the details may be slightly different.



The main window consists of the following items:

- **Toolbar:** contains short cuts to several commonly used items
 - The simulation tool: shaped like a hand, is used in simulation mode to alter input pins.
 - The design tool: is used while designing the circuit.
 - The input pin: green circle surrounded by a square box, is used to send a signal through a wire. When placing the input on the canvas it initializes the input to logic 1 or 0. The number of bits can be increased in the Attribute Table.
 - The output pin: green circle surrounded by a circular shape, is used to observe the output from a gate or a block. The output pin toggles in real time as long as the simulation is enabled from the menu bar: **Simulate > Simulation Enabled**
- **Explorer Pane:** This pane contains the list of wiring, gates, multiplexers and other components that are available for digital design in Logisim.
- **Attribute Table:** Gives detailed attributes of digital design components (e.g., AND, OR, XOR gates). The attribute table allows you to alter the number of inputs/outputs that a digital component may have.
- **Canvas:** The canvas is the area for you to create your digital circuits. In this area you may simulate your circuits while designing in real time.

Circuit Design:

Logisim operates in two modes: Design and Simulate. Logisim is first operated in edit mode while designing a logic circuit, then in simulate mode to see how the circuit would actually work.

- **Edit Mode:**
- To use the edit mode, select the **Design** tool
- Select then a component in the library on the left. To add it in your design, simply drag and drop the desired component in the canvas:
 - To insert a logic gate, expand the folder Gates and click on the desired logic gate. Once the gate is selected, the mouse will turn into the shape of the selected gate. Place the gate in the canvas or circuit area on the right.
 - Each selected component has modifiable attributes in the attribute area. For example for an AND gate, the number of input signals, the number of bits per input or output and the size of the component can all be modified.
 - It is also possible to copy and paste of one or more components in the canvas. In this case, the components retain all the attributes previously defined.
- To connect the gates, select the arrow icon on top. Then drag from the output of one gate to the input of another gate. You can connect to any of the small blue dots on the gate symbol.
- To create an input to the circuit, select the “**Add Pin**” icon with an outline of a *square*.
- Similarly, add an output to the circuit by using the “**Add Pin**” icon with an outline of a *circle*.
- To assign a name to the input/output pin, click on the pin while the arrow icon is selected.
- You may then add the label for the pins.
- While components are added to the circuit, notice how the color of the wire changes

Logisim Wire Color Conventions:

Wire Color	Meaning
dark green	logical '0'
light green	logical '1'
blue	unknown value
gray	unconnected wire
black	wire has more than one bit (bus)
red	conflicting values, error
orange	incompatible bus width

Example Circuit:**Full Adder:**

In Logisim, a digital circuit may be designed using either one of two approaches:

- Drawing schematic: the user has to do the design by choosing the necessary components as per his design, based on the digital schematic.
- Analyzing the circuit, by providing the truth tables with necessary inputs and outputs and let Logisim do the design.

Schematic Design:

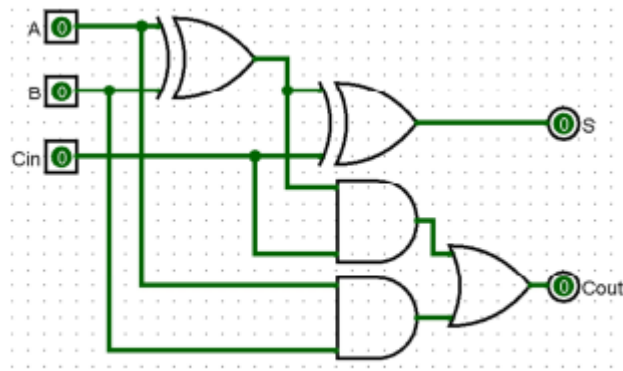
- a. Set up the truth table for a simple Full Adder. Let A B and Cin be the inputs, S and Co the outputs.

A	B	Ci	S	Co
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

- b. Derive the logic equations for S and Cout:

$$S = A \oplus B \oplus C_i \quad C_o = (A \wedge B) + (C_i \wedge (A \oplus B))$$

- c. Finally, design the circuit based on the above equations



Full Adder Circuit Design

Circuit Analysis:

From the Menu Bar select "**Window**" then "**Combinational Analysis**". You may also go to **Project** then, select "**Analyze Circuit**".

- Create three inputs named Cin, A, and B.
- Create two outputs named Cout and S.
- Complete the truth table for a full adder.
- Look at the "**Expression**" and "**Minimized**" tabs.
- Press the "**Build Circuit**" button and enter FA0 as the name of the new circuit.

Once your design is complete, you may proceed by testing its functionality:

- Use the Simulation Tool in the Toolbar.
- Click on an input to see its state toggle between the values 0 and 1
- While doing so, check the outputs that they correspond exactly to your design.

Test Mode:

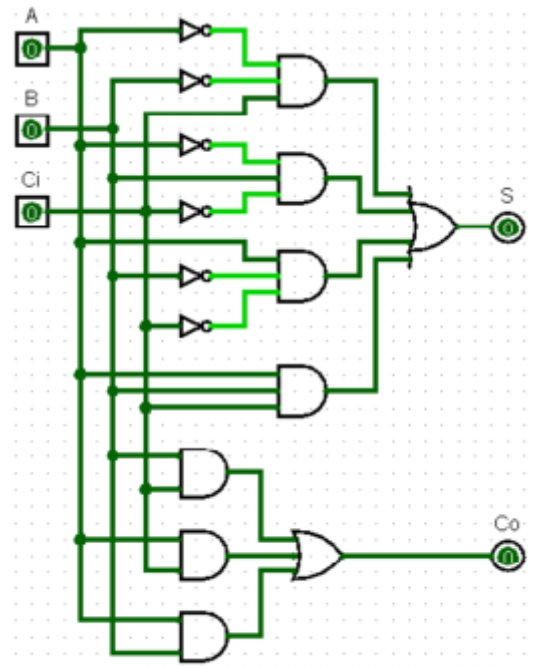
Select the Simulation Tool (Figure: 10.1) so you can test the circuit. You can toggle the value of an input pin by clicking on it. Note that the values of all subsequent wires change instantly.

- Change the values of the Ci, A, and B inputs
- Observe the Co and S outputs to verify the correct operation of the circuit

Probes

As you test your circuit you may also add probes to check the values on wires or buses. These values may be displayed in different formats.

- From the explorer pane, select Wiring then Probe.
- Set a direction for your probe,
- Then select the numbering system that you would like to use to display your data.



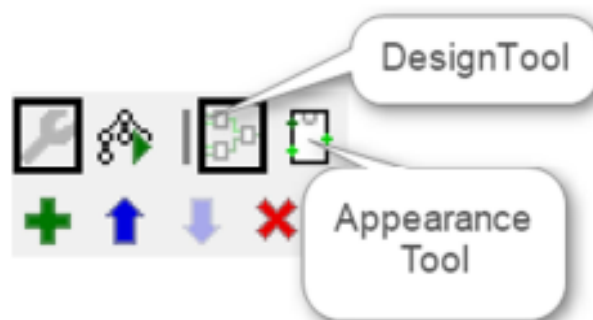
Full Adder Circuit Designed Through Analysis

Hierarchical Design:

A complex circuit is usually broken down into smaller sub-circuits. These sub-circuits are designed independently, tested, arranged into libraries, and then used to build the big circuit.

Circuit Appearance:

Later on, you may use your circuit as a **sub-circuit** in another design. For that purpose, you may want to alter the locations of the inputs and outputs or the appearance of the whole circuit as a module. To do so, while your design is on the editor, click the Appearance Tool under the toolbar. This will show your circuit as a block with the inputs and outputs; once you click any of the input/output pins, on the bottom right corner your circuit will appear showing the location of the pin you selected.



You may then modify the following properties of your circuit appearance: pin locations, general appearance, and color.

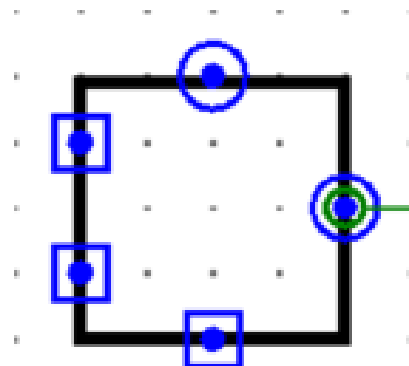
Using Sub Circuits:

One of the good features of Logisim is that you can use a circuit you have already built as a building block in another, more complex circuit.

You can create a new sub-circuit with (Project > Add Circuit)

- Save all your sub-circuits in one folder that you will make your own library.
- On the menu bar select load Project > Load Library > Logisim-Library then select your folder and the module you would like to add.
- Your circuit will appear as a new element in the explorer pane. Select your module then drag and drop.
- You may then add pins for input and output, or connect your module to your circuit using wires.

Within the newly created full adder FA0 circuit designed above (Figures 10.2 or 10.3), change the orientation of the Ci input so that it is facing south. Change the orientation of the Co output so that it is facing north (Figure 10.5). Your circuit is ready to be used as a module or sub-circuit in a more complex digital system design.



Full Adder Block

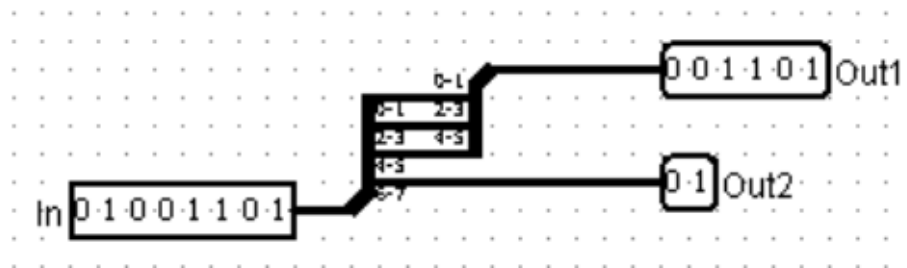
Splitters & Tunnels:

These are components mainly used to simplify wiring. Splitters are used to group or separate bits in a bus, and tunnels are used to avoid lengthy connections between components in the design.

Splitters:

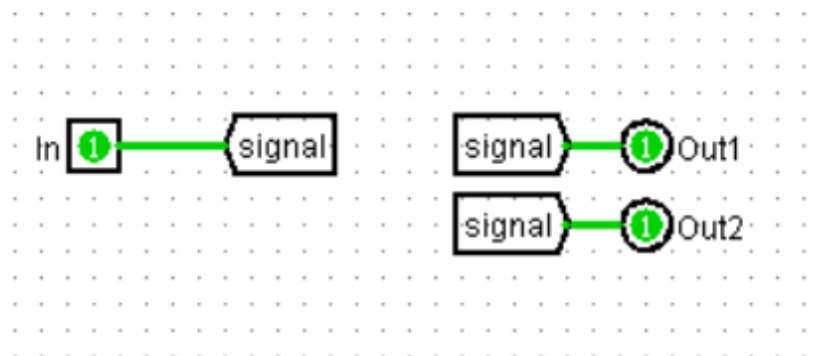
Wire splitters split a multi-bit wire into smaller wires (an 8-bit wire into 2 x 4-bit wires, for example). They can also combine multiple wires into a single wire with a wider bit width.

- To use a splitter from the menu select (Wiring > Splitter)
- Set the fan out and the bit width in properties.
 - "Fan Out" controls how many teeth the wire splitter has.
 - "Bit Width In" controls the actual size of the data the splitter is splitting.
- If you have more data bits than outputs, some of the fan outs would carry more than one data bit.



Tunnels:

Tunnels are you to transmit signals without using wires.



- From the menu, select (Wiring > Tunnel)
- Tunnels are mainly used to avoid excessive wiring, but can be used to connect control and clock signals.

Lab Tasks

- Design & Implement Encoder & Decoder on Logisim **(02 Marks)**
- Design & Implement Full Adder & Full Subtractor on Logisim **(02 Marks)**
- Design & Implement 4 x 1 MUX (3 bits) on Logisim **(02 Marks)**

Observations:

(02 Marks)

Introduction to Latches

EXPERIMENT NO. 11

Name	Report Marks (05)	Lab Performance (05)	Viva Marks (05)	Total (15)

EXPERIMENT NO. 11

Introduction to Latches

Objectives

- Introduction to Latched .
- Implementation of SR, JK, D & T Latches.

Components

- ICs – 74LSxx
- Wire Stripper
- Prototyping board with power and ground connections

Theory:

Latches:

Latches are asynchronous – which means, the output of the latch depends on its input; on the other hand, today, most computers are synchronous – which means, the outputs of all the sequential circuits change simultaneously to the rhythm of a global clock signal. There are four types of latches: D, T, SR and JK latch.

SR Latch:

An SR latch (Set/Reset) is an asynchronous device: it works independently of control signals and relies only on the state of the S and R inputs. In the image we can see that an SR latch can be created with two NOR gates that have a cross-feedback loop. SR latches can also be made from NAND gates, but the inputs are swapped and negated. In this case, it is sometimes called SR latch.

When a high is applied to the Set line of an SR latch, the Q output goes high (and Q low). The feedback mechanism, however, means that the Q output will remain high, even when the S input goes low again. This is how the latch serves as a memory device. Conversely, a high input on the Reset line will drive the Q output low (and Q high), effectively resetting the latch's "memory". When both inputs are low, the latch "latches" – it remains in its previously set or reset state.

When both inputs are high at once, however, there is a problem: it is being told to simultaneously produce a high Q and a low Q. This produces a "race condition" within the circuit - whichever flip flop succeeds in changing first will feedback to the other and asserts itself. Ideally, both gates are identical and this is "metastable", and the device will be in an undefined state for an indefinite period. In real life, due to manufacturing methods, one gate will always win, but it's impossible to tell which it will be for a particular device from an assembly line. The state of $S = R = 1$ is therefore "illegal" and should never be entered.

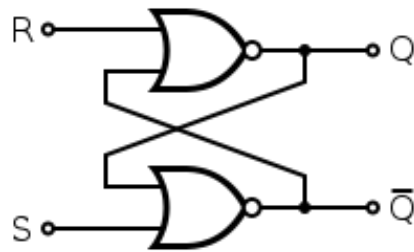


Fig 11.1: SR Latch using NOR logic

S	R	Q	Q'
1	0	0	1
1	1	0	1 (after $S = 1, R = 0$)
0	1	1	0
1	1	1	0 (after $S = 0, R = 1$)
0	0	1	1 (forbidden)

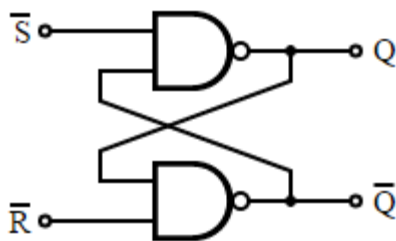
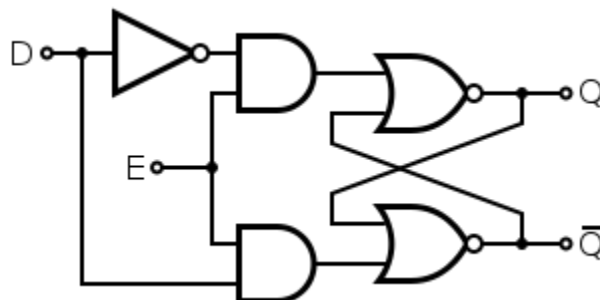


Fig 11.2: SR Latch using NAND logic

D Latch:

The D latch (D for "data") or transparent latch is a simple extension of the gated SR latch that removes the possibility of invalid input states. Since the gated SR latch allows us to latch the output without using the S or R inputs, we can remove one of the inputs by driving both the Set and Reset inputs with a complementary driver: we remove one input and automatically make it the inverse of the remaining input.

The D latch outputs the D input whenever the Enable line is high, otherwise the output is whatever the D input was when the Enable input was last high. This is why it is also known as a transparent latch – when Enable is asserted, the latch is said to be "transparent" - it signals propagate directly through it as if it isn't there.

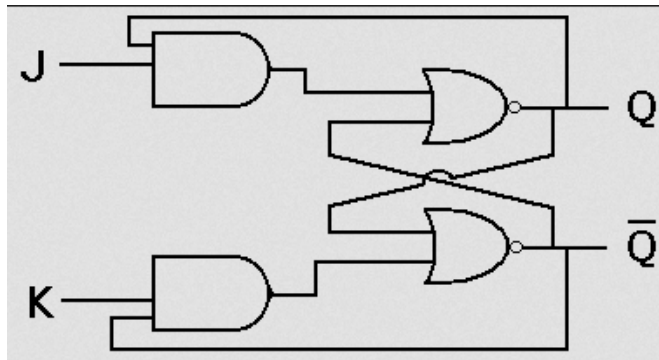


En	D	Next state of Q
0	X	No change
1	0	$Q = 0$; reset state
1	1	$Q = 1$; set state

Fig 11.3: D-Latch

J-K Latch:

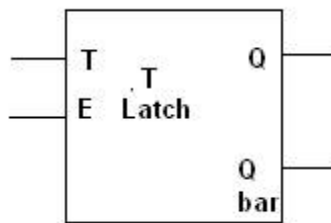
JK latch is similar to RS latch. This latch consists of 2 inputs J and K as shown in the below figure. The ambiguous state has been eliminated here: when the inputs of Jk latch are high, then output toggles. The output feedback to inputs is the only difference we see here, which is not there in the RS latch.

**Fig 11.4: JK-Latch**

J	K	Previous Q	Current Q
0	0	0	0 (hold state)
0	0	1	1 (hold state)
0	1	X (don't care)	0 (reset)
1	0	X	1 (set)
1	1	0	1 (toggle)
1	1	1	0 (toggle)

T-Latch:

T latch is formed when the inputs of the JK latch are shorted. When the input is high, then the output toggles.

**Fig 11.5: T-Latch**

T	Previous Q	Current Q
0	0	0
0	1	1
1	0	1
1	1	0

Lab Task

- Construct the circuits of SR, JK, D & T Latches on Breadboard and verify the truth tables.

(04 Marks)**S-R Latch:****(01 Mark)****J-K Latch:****(01 Mark)****D Latch:****(01 Mark)**

T-Latch:**(01 Mark)****Observations:****(01 Mark)**

Review Questions

- 1) Differentiate Latch & Flip-Flop? (01 Mark)

- 2) How can you differentiate between SR and D latch? (01 Mark)

- 3) How can you define sequential circuits? Is there any difference between sequential and combinational logic? (01 Mark)

- 4) What happens when both inputs of SR latch go high? Explain your answer with comprehensive details. (01 Mark)

Introduction to Flip-Flops

EXPERIMENT NO. 12

Name	Report Marks (05)	Lab Performance (05)	Viva Marks (05)	Total (15)

EXPERIMENT NO. 12

Introduction to Flip-Flops

Objectives

- Introduction to Flip Flops.
- Implementation of SR, JK, D & T Flip-Flops.

Components

- ICs – 74LSxx
- Wire Stripper
- Prototyping board with power and ground connections

Theory:

Flip – Flops:

A flip flop can be designed by using two NOR gates or two NAND gates. A basic flip flop using NAND gate is shown below. Each flip flop has two inputs set and reset and also two outputs Q and Q'. This type of flip flop is referred to as an SR flip flop or SR latch. The flip-flop has two states which are shown in the below figure. When $Q=1$; and $Q'=0$; it is in the set state. When $Q=0$ & $Q'=1$, it is in the clear state. The outputs of the flip flop Q & Q' are complements of each other and are referred to as the normal and complement outputs, respectively. The flip flop binary state is taken to be the value of the normal output. When 1 is applied to the inputs of the flip flop, both the outputs go to 0, so both the outputs are complements of each other. In a normal operation, this condition must be avoided by making sure that 1's are not applied to both the inputs simultaneously.

SR Flip Flop

This SR flip-flop consists of two AND gates and a basic NOR flip-flop. The outputs of the two AND gates remain at 0 as long as the clock pulse is 0, irrespective of the input values of S & R. When the clock pulse is 1, information from the inputs S & R passes through to the basic flip-flop. When $S=R=1$, the occurrence of a clock pulse causes both the outputs go to 0. When the clock pulse is removed, the state of the flip-flop is unstated.

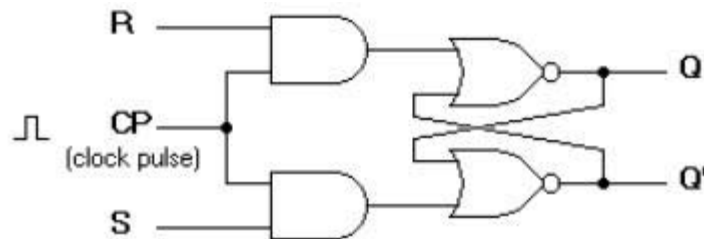


Fig 12.1: S R Flip Flop

Q	S	R	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	indeterminate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	indeterminate

D Flip Flop:

The D flip-flop is the modification of the SR flip flop which is shown in the figure. The i/p D goes directly into the input S and the complement of the input D goes to the input R. The D input is sampled during the existence of a clock pulse. If it is 1, then the flip-flop is switched to the set state. If it is 0, then the flip-flop switches to the clear state.

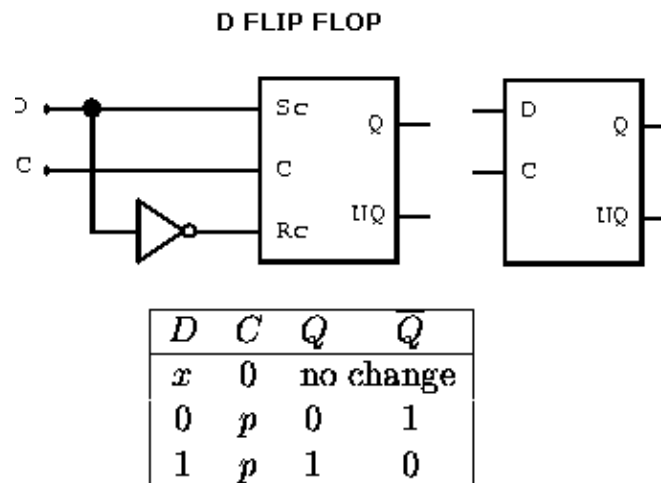


Fig 12.2: D Flip Flop

JK Flip Flop:

A JK flip flop is a modification of the SR flip flop. The inputs of the flip flop J, K behave like the inputs S and R. When input 1 is applied to both J & K, the flip flop switches to its complement state (if $Q=1$, it switches to $Q=0$). The JK flip flop figure is shown below. The output of the flip flop Q is ANDed with inputs k and clock pulse. The flip flop would be cleared during a clock pulse only if the output Q was previously 1. Likewise, the output Q' is ANDed with inputs CP and J. So that the flip flop is set with a clock pulse only if Q' was previously 1

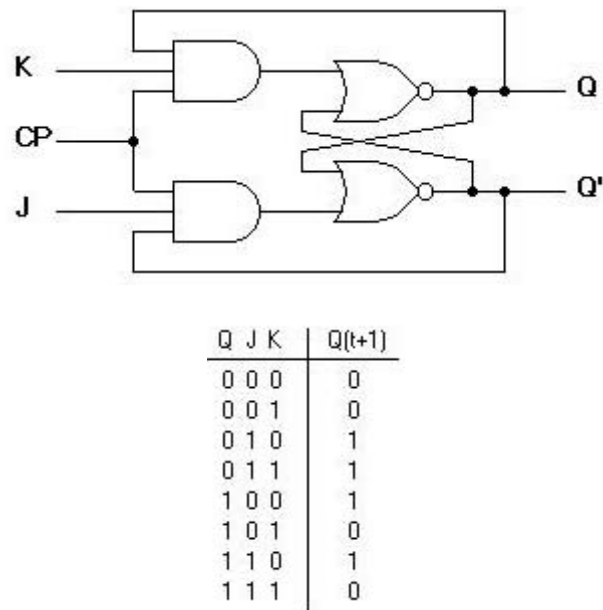


Fig 12.3: J K Flip Flop

T Flip Flop:

The T flip flop is a single input version of the JK flip flop. The operation of this T flip flop is as follows: When the input of the T is '0' such that the 'T' will make the next state the same as the present state (i.e. T = 0 then, present state = next state = 0). However, if the input of the T is '1' then the 'T' will change the next state to the inverse of the present state (i.e. T = 1 present state = 0 and next state = 1).

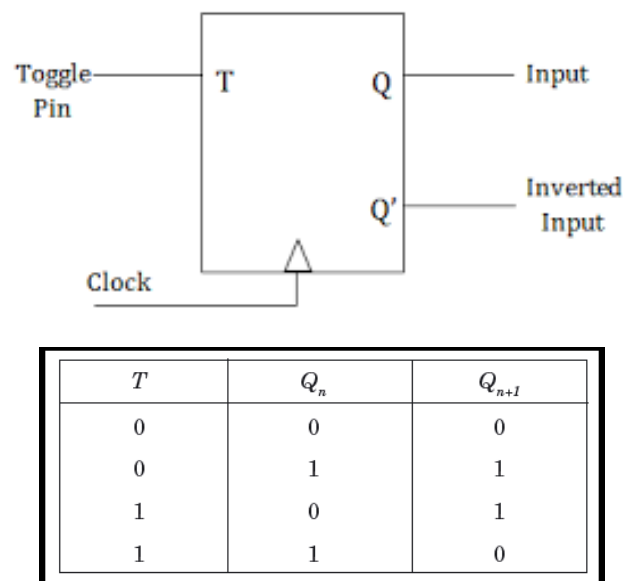


Fig 12.4: T- Flip Flop

Lab Task

- Construct the circuits of SR, JK, D & T Flip Flops on Breadboard and verify the truth tables

(04 Marks)

S-R Flip Flop:

(01 Mark)



J-K Flip Flop:

(01 Mark)



D Flip Flop:

(01 Mark)



T- Flip Flop:**(01 Mark)****Observations:****(01 Mark)**

Review Questions

- 1) Why a flip flop is always triggered by a clock is it essential if so give reasons? (01 Mark)

- 2) What is a transparent flip flop? (01 Mark)

- 3) What does the triangle on the clock input of a JK flip flop mean? (01 Mark)

- 4) How can JK flip flop overcome race around condition? (01 Mark)

Counters

EXPERIMENT NO. 13

Name	Report Marks (05)	Lab Performance (06)	Viva Marks (04)	Total (15)

EXPERIMENT NO. 13

Counters

Objectives

- Introduction to Counters
- Implementation of Synchronous & Asynchronous Counters.

Components

- ICs – 74LSxx
- Wire Stripper
- Prototyping board with power and ground connections

Theory:

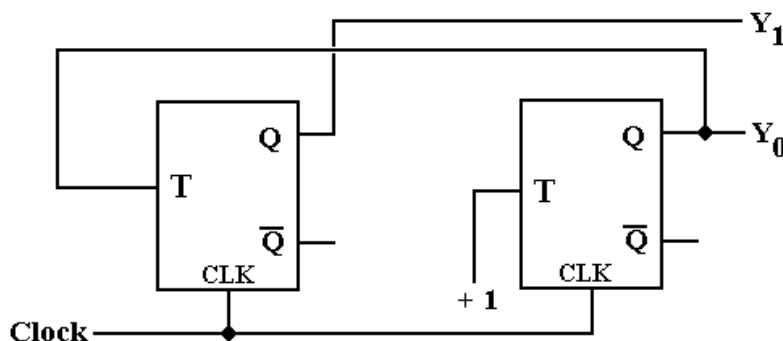
A *counter* is a sequential logic circuit that goes through a prescribed sequence of states upon the application of input pulses. The prescribed sequence can be a binary sequence or any other sequence. A counter that goes through 2^N (N is the number of flip-flops in the series) states is called a *binary counter*. The modulus of a counter is the number of different states it is allowed to have. Counter modulus is normally 2^N unless controlled by a feedback circuit which limits the number of possible states (an example being the decimal counter). Counters are very widely used in almost all computers and other digital electronic systems. There are two major categories of counters: asynchronous counters and synchronous counters.

There are two types of Counters

- Synchronous Counter.
- Asynchronous Counter.

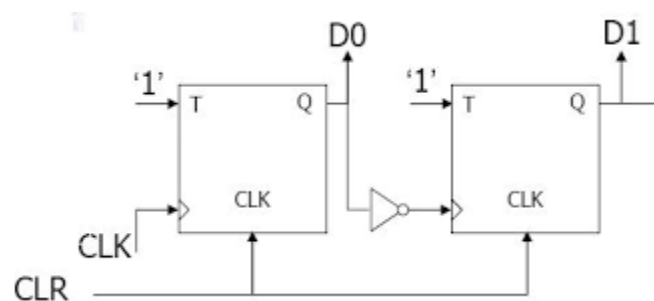
Synchronous Counter:

Synchronous Counters are so called because the clock input of all the individual flip-flops within the counter are all clocked together at the same time by the same clock signal.



Asynchronous Counter:

Counters arranged so that the output of one flip-flop generates the clock input of the next higher stage are generally called *asynchronous counters* (or ripple counter). In other words, in asynchronous counters, the CLK inputs of all flip-flops (except the first one) are triggered not by the incoming pulses but rather by the transition that occurs in other flip-flops. Therefore, the change of state of a particular flip-flop is dependent upon the present state of other flip-flops. Fig. 1 shows a count-up ripple counter. When a transition from, say, 0111 to 1000 occurs, the one-to-zero transition of the low-order three bits ripples from bit to bit. Since each flip-flop has a non-zero propagation delay, ripple counters are relatively slow. Therefore, an upper limit on the number of flip-flops in the flip-flop chain ought to be imposed.



Types of Synchronous & Asynchronous Counter:

- UP Counter.
- DOWN Counter.
- Up/Down Counter.
- Decade Counter.
- Ring Counter.

How to Design Synchronous Counter:

Step1: Counter type, Bits & Flip-Flop type.

Step2: Decode the number of Flip-Flop.

Step3: Excitation Table of Flip-Flop.

Step4: State diagram and Circuit excitation table.

Step5: Obtain simplified equation using K-Map

Step6: Draw the logic diagram.

Example:

Step 1: To design a synchronous up counter, first we need to know what number of flip flops are required. we can find out by considering number of bits mentioned in question. So, in this we required to make 3 bit counter so the number of flip flops required are 3 [2^n where n is number of bits].

Step 2: After that, we need to construct state table with excitation table. Note: To construct excitation table from state table you should know the excitation table of respective flip flop in this case it is T flip flop. So check the excitation table for T flip flop Which is:

T Flip Flop Excitation Table

Present state	Next State	T
0	0	0
0	1	1
1	0	1
1	1	0

So, the above table is excitation table for T Flip Flop.

State Table with excitation table

Present State			Next State			Flip Flop		
Q_3	Q_2	Q_1	Q_3'	Q_2'	Q_1'	T_3	T_2	T_1
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

Above table is created as per follow :

When $Q_3 = 0$ which is present state and $Q_3' = 0$ which is next state then T_3 become 0 [As per excitation table, have a look]

Similarly, if Q_3 is 0 and Q_3' is 1 then T_3 become 1.

In similar way it goes on .

Step 3: After making the excitation table the next thing to do is dig out the equation from the boolean algebra or K map for the design of the counter. So, for T_1 , T_2 and T_3 we got 1, Q_1 and $Q_1.Q_2$

K-Map

For T_3 Flip flop,

		Q3 Q2			
Q1	0	0	0	0	
	0	1	1	0	

$$T_3 = Q_1.Q_2$$

For T_2 Flip flop,

		Q3 Q2			
Q1	0	0	0	0	
	1	1	1	1	

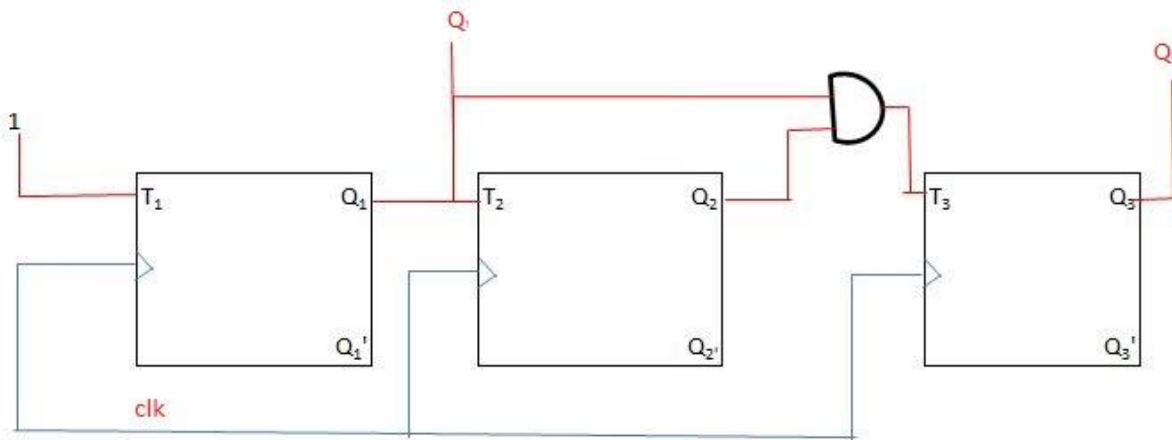
$$T_2 = Q_1$$

For T_1 Flip flop,

		Q3 Q2			
Q1	1	1	1	1	
	1	1	1	1	

$$T_1 = 1$$

Step 4: Lastly according to the equation got from K map create the design for 3 bit synchronous up counter.



In above design T_1 is getting input 1 and T_2 is getting input from output of the T_1 flip flop and lastly, T_3 is getting input from the output of T_1 and T_2 . A clock is attached to it which is in blue color.

How to Design Asynchronous Counter:

Step1: Counter type, Bits & Flip-Flop type.

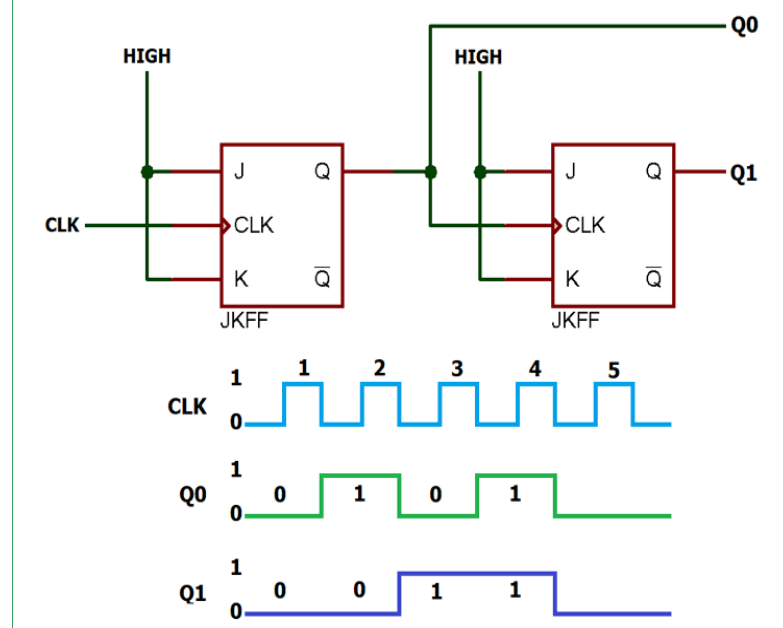
Step2: Decide the number of Flip-Flop.

Step3: Waveform.

Step4: State diagram and Circuit excitation table.

Step5: Truth table (Clock pulse, Outputs, decimals)

2-Bit Asynchronous Counter

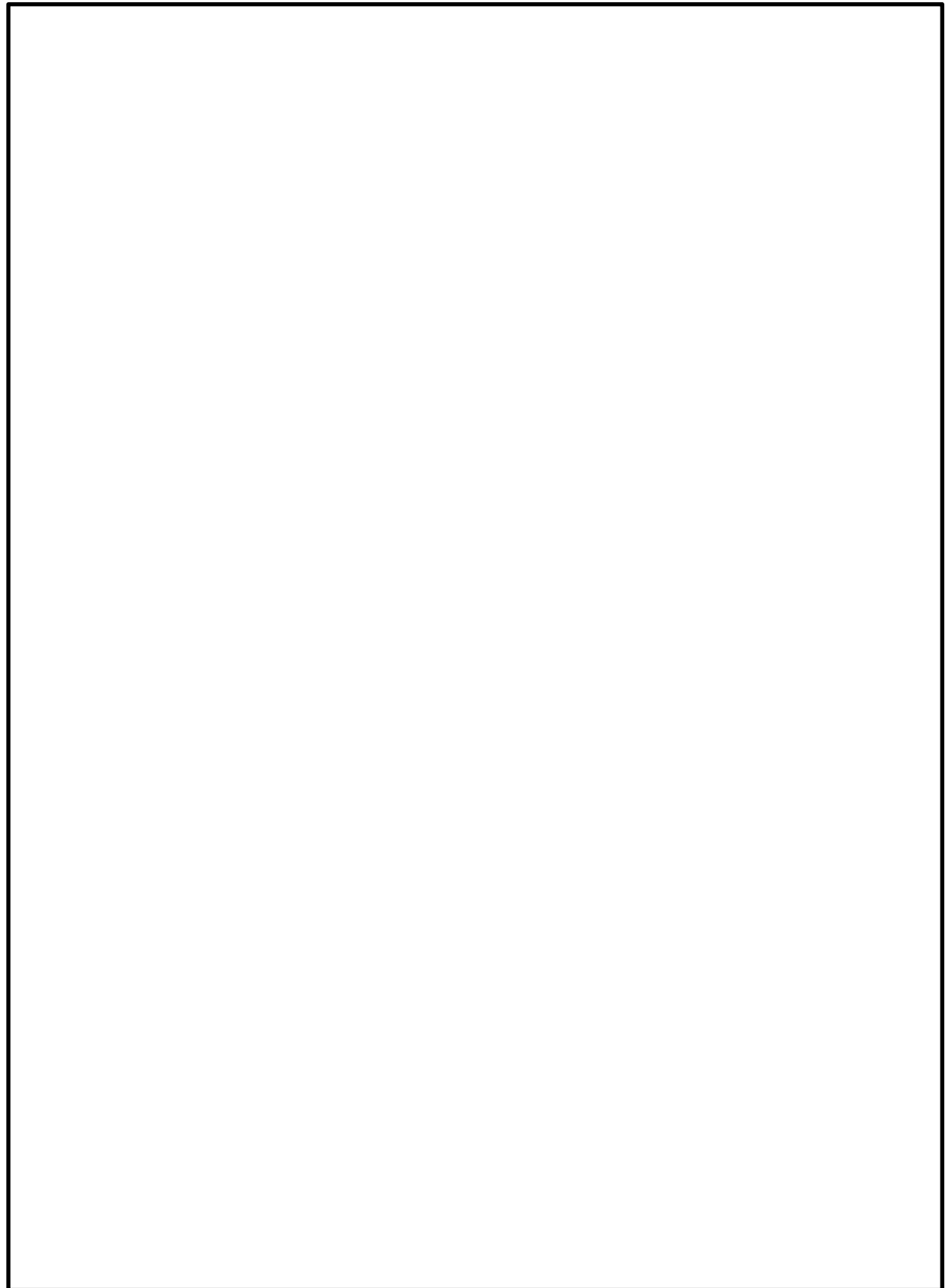


<u>Clock</u>	<u>Outputs</u>		<u>Decimal</u>
	Q1	Q0	
0	0	0	0
1	0	1	1
2	1	0	2
3	1	1	3

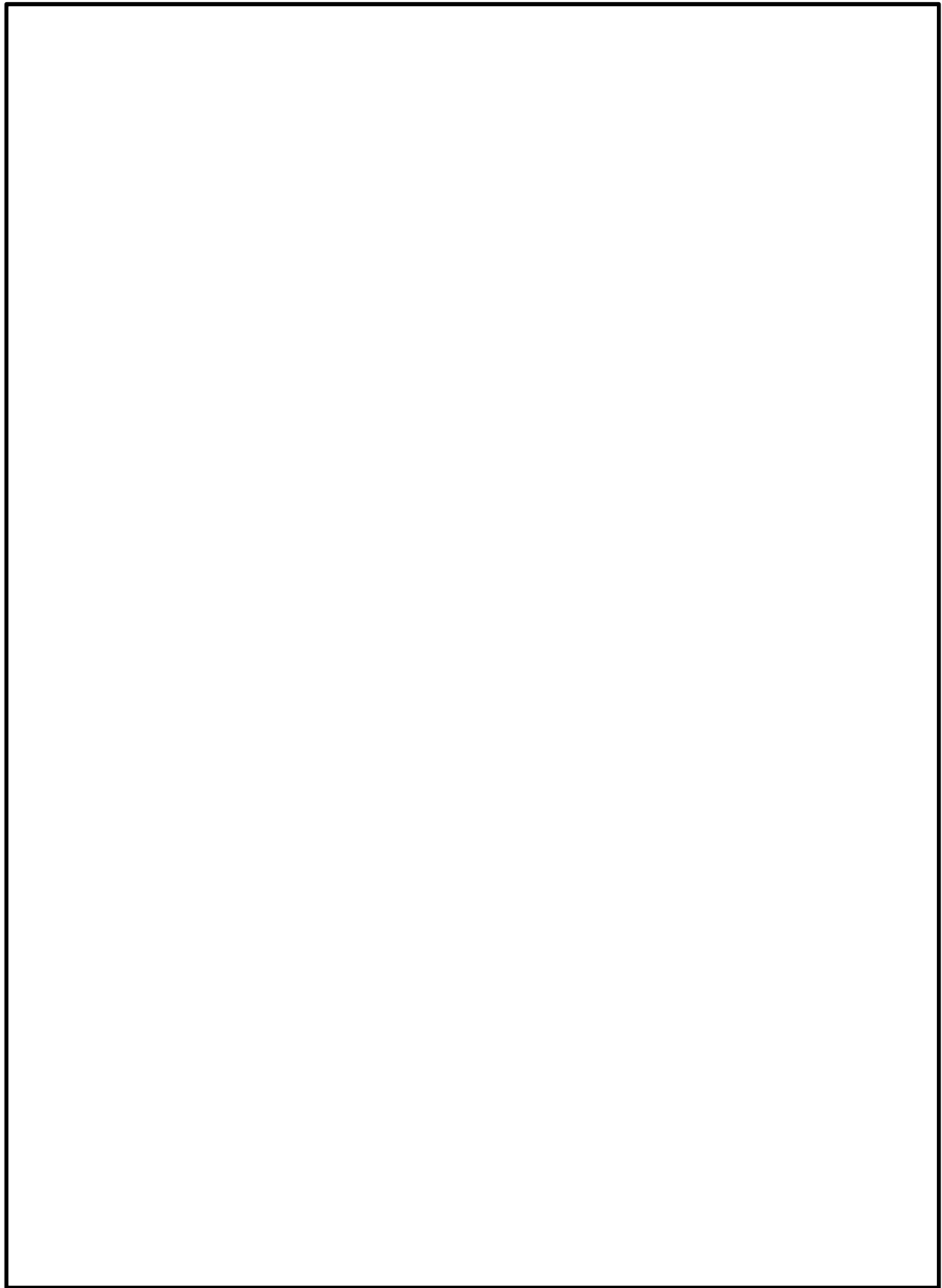
LAB TASK

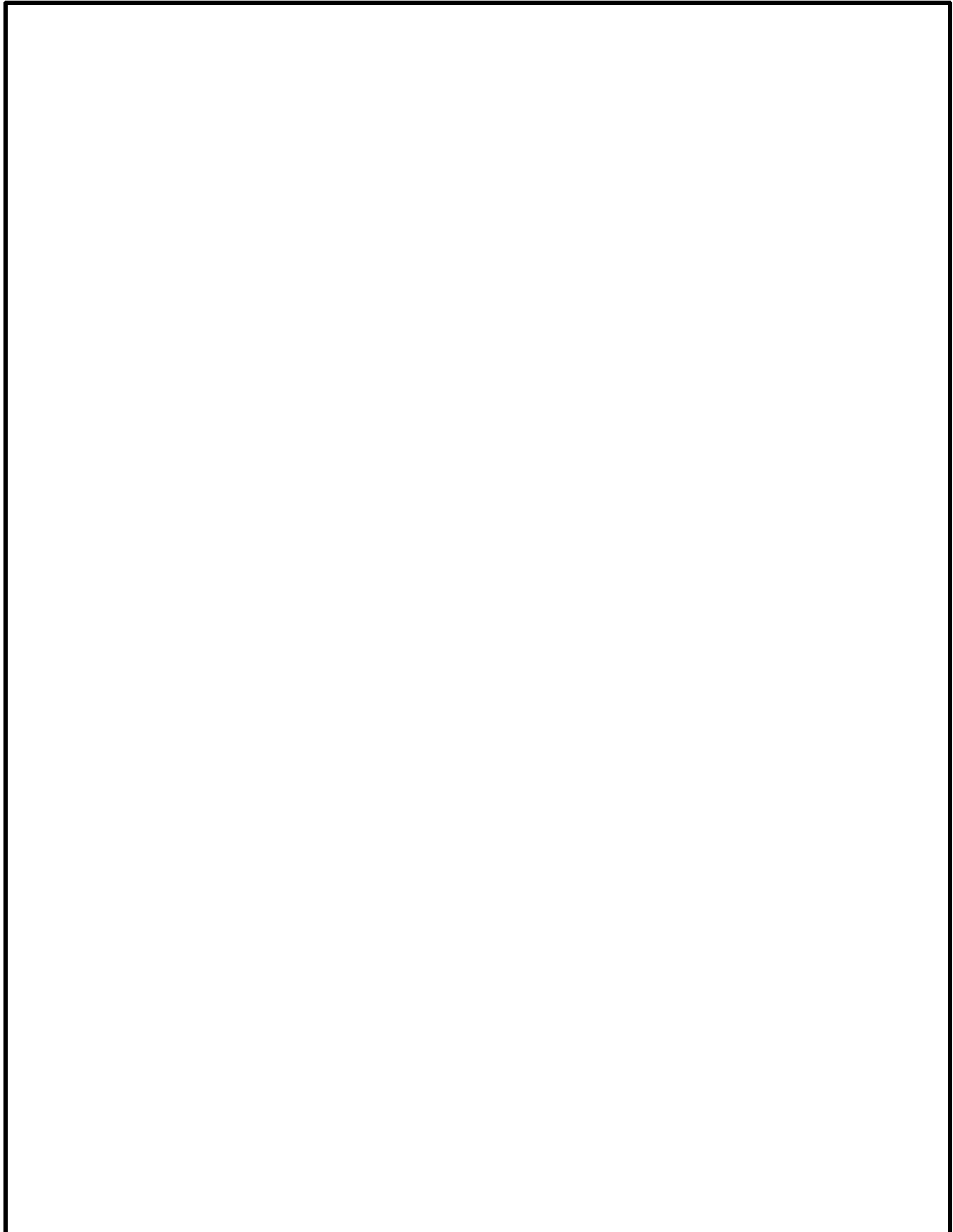
- Design 2 bits Synchronous up Counter using J-K Flip-Flop (02 Marks)
- Design 3 bits Asynchronous up Counter using J-K Flip-Flop (02 Marks)
- Design 3 bits Asynchronous down Counter using J-K Flip-Flop (02 Marks)

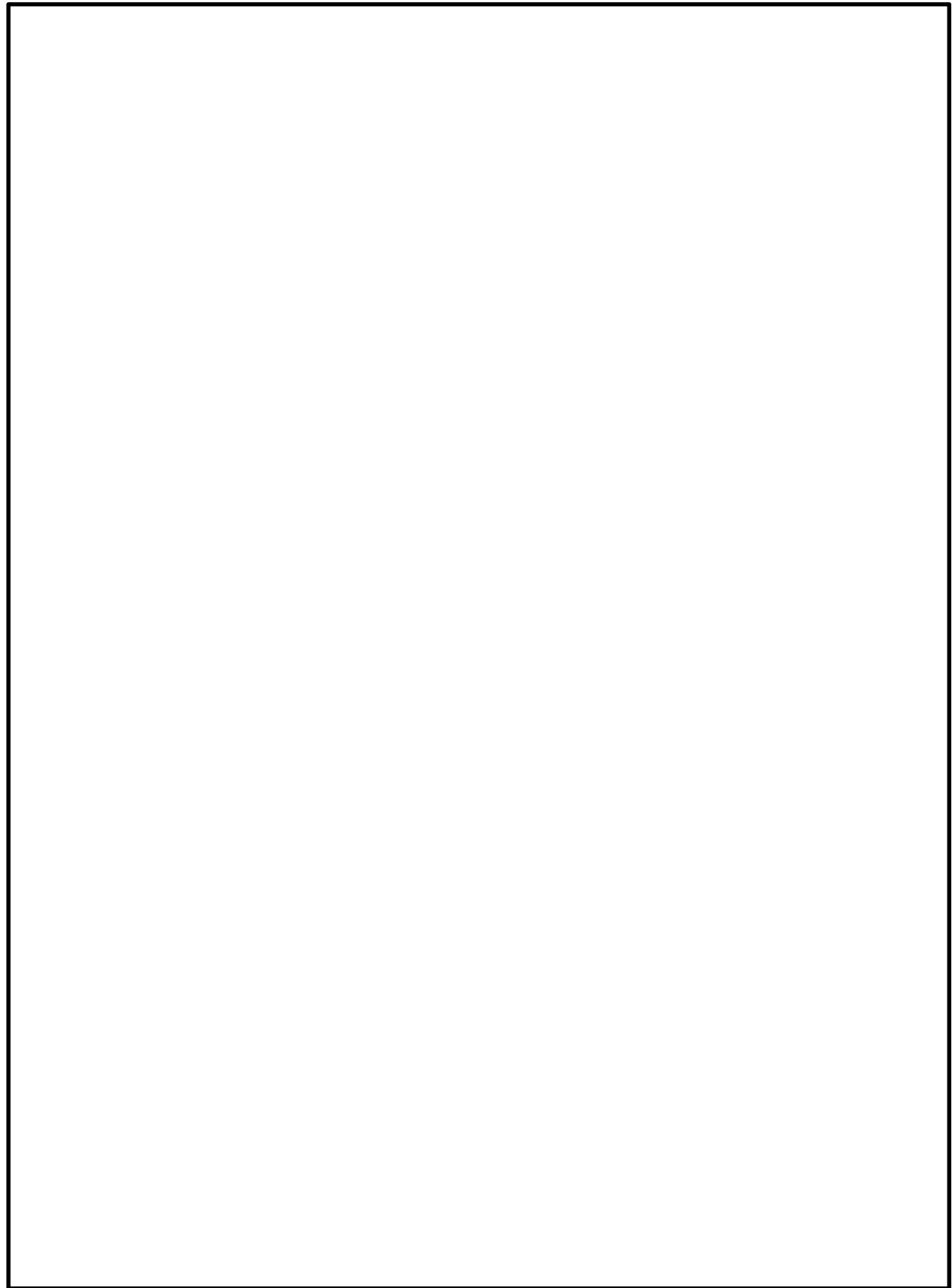
Task 1**Design 2 bits Synchronous up Counter using J-K Flip-Flop**



Task 2**Design 3 bits Asynchronous up Counter using J-K Flip-Flop**



Task 3**Design 3 bits Asynchronous down Counter using J-K Flip-Flop**



Observations:

(01 Mark)

Review Questions

- 1) Differentiate Synchronous & Asynchronous Counter? (01 Mark)

- 2) Advantage & Disadvantages of Synchronous Counter? (01 Mark)

- 3) Advantages & Disadvantages of Asynchronous Counter (01 Mark)

- 4) What is Clock pulse? (01 Mark)

Shift Registers

EXPERIMENT NO. 14

Name	Report Marks (05)	Lab Performance (08)	Viva Marks (02)	Total (15)

EXPERIMENT NO. 14

Shift Registers

Objectives

- Introduction to Shift Registers.
- Implementation of SISO, SIPO, PISO, PIPO

Components

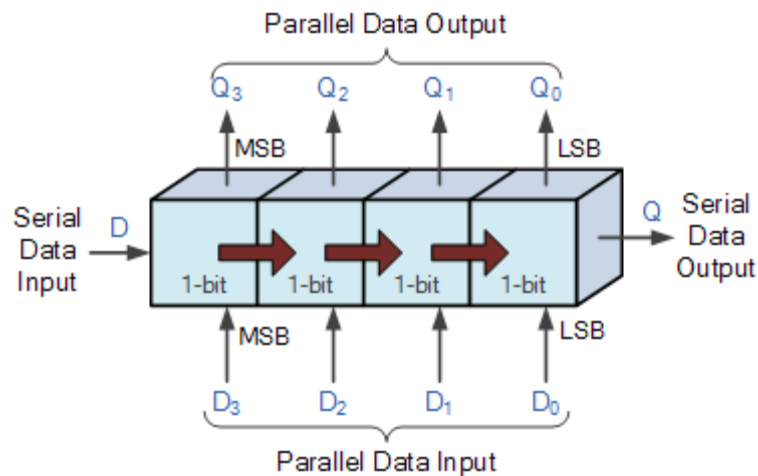
- ICs – 74LSxx
- Wire Stripper
- Prototyping board with power and ground connections

Theory:

- The Shift Register is another type of sequential logic circuit that can be used for the storage or the transfer of binary data
- This sequential device loads the data present on its inputs and then moves or “shifts” it to its output once every clock cycle, hence the name Shift Register.
- A shift register basically consists of several single bit “D-Type Data Latches”, one for each data bit, either a logic “0” or a “1”, connected together in a serial type daisy-chain arrangement so that the output from one data latch becomes the input of the next latch and so on.
- Data bits may be fed in or out of a shift register serially, that is one after the other from either the left or the right direction, or all together at the same time in a parallel configuration.
- The number of individual data latches required to make up a single Shift Register device is usually determined by the number of bits to be stored with the most common being 8-bits (one byte) wide constructed from eight individual data latches.
- Shift Registers are used for data storage or for the movement of data and are therefore commonly used inside calculators or computers to store data such as two binary numbers before they are added together, or to convert the data from either a serial to parallel or parallel to serial format. The individual data latches that make up a single shift register are all driven by a common clock (Clk) signal making them synchronous devices.
- Shift register IC’s are generally provided with a clear or reset connection so that they can be “SET” or “RESET” as required. Generally, shift registers operate in one of four different modes with the basic movement of data through a shift register being
- **Serial-in to Parallel-out (SIPO):** The register is loaded with serial data, one bit at a time, with the stored data being available at the output in parallel form.
- **Serial-in to Serial-out (SISO):** The data is shifted serially “IN” and “OUT” of the register, one bit at a time in either a left or right direction under clock control.
- **Parallel-in to Serial-out (PISO):** The parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.
- **Parallel-in to Parallel -out (PIPO):** The parallel data is loaded simultaneously into the

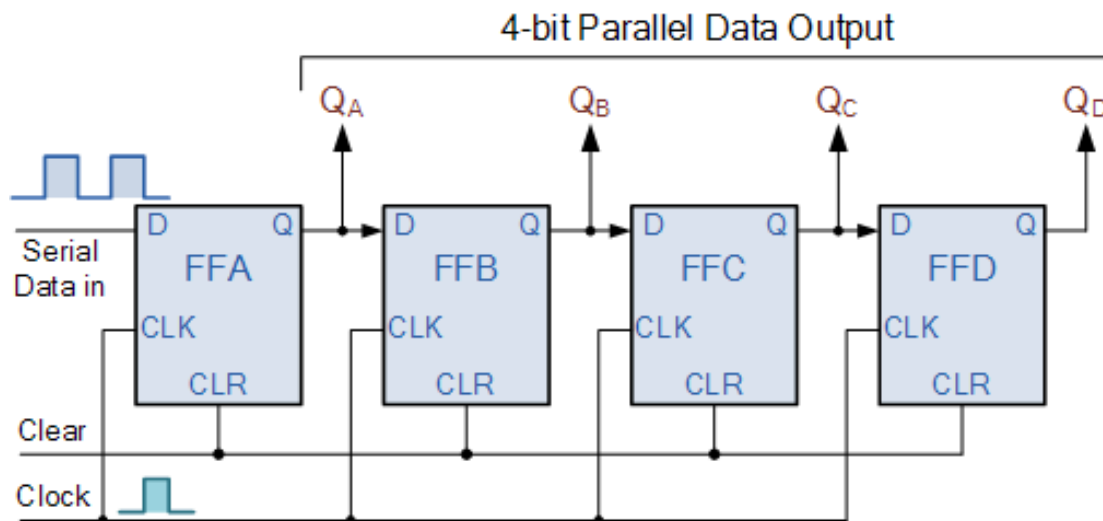
register, and transferred together to their respective outputs by the same clock pulse.

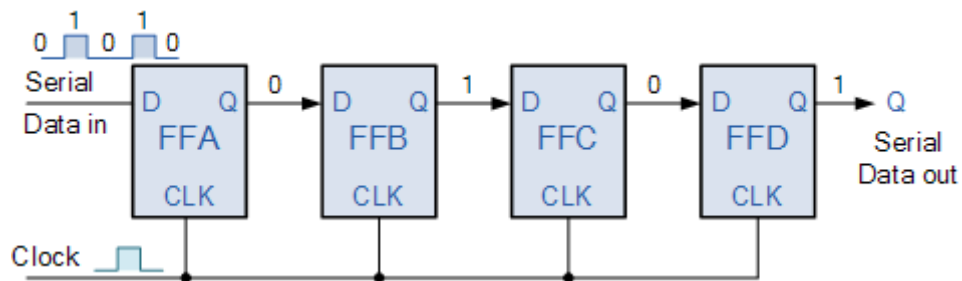
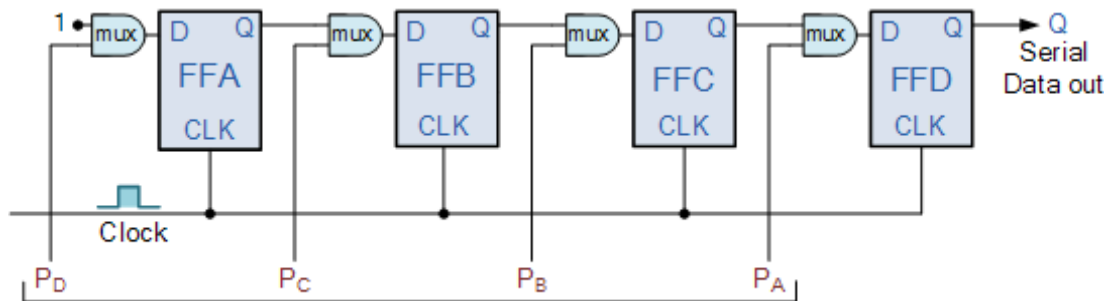
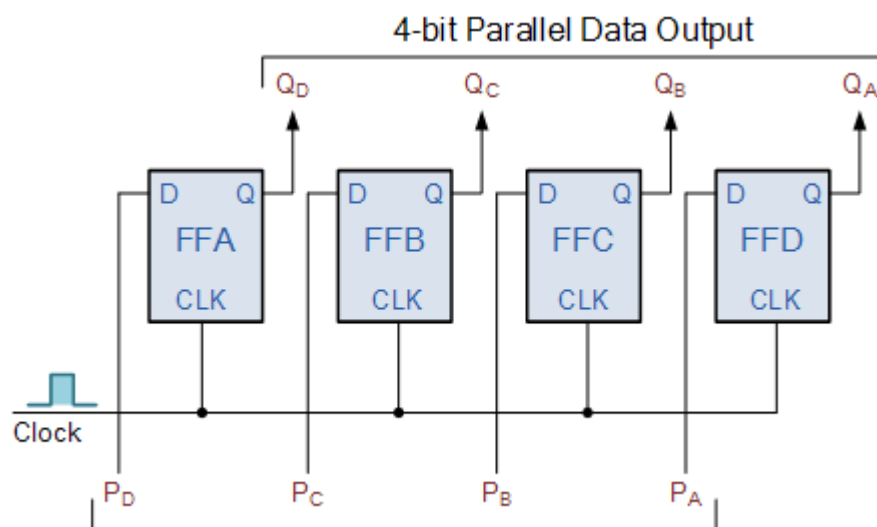
- The effect of data movement from left to right through a shift register can be presented graphically as:

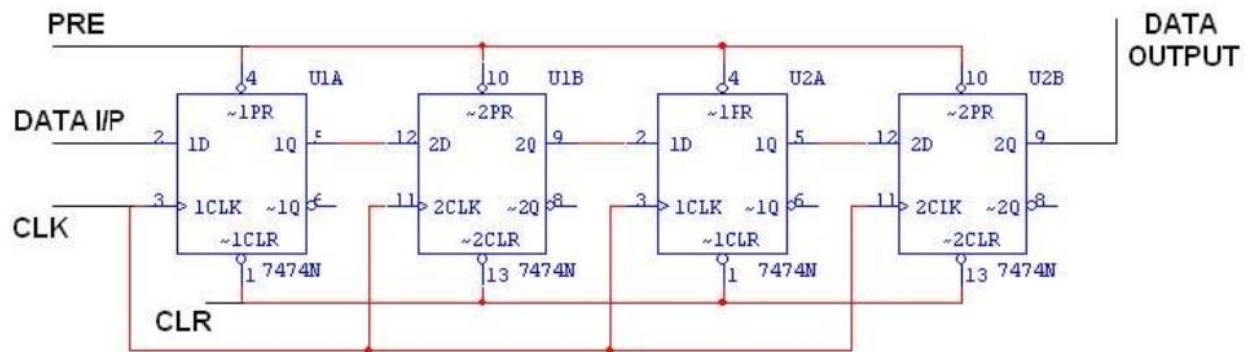


Also, the directional movement of the data through a shift register can be either to the left, (left shifting) to the right, (right shifting) left-in but right-out, (rotation) or both left and right shifting within the same register thereby making it *bidirectional*. In this tutorial it is assumed that all the data shifts to the right, (right shifting).

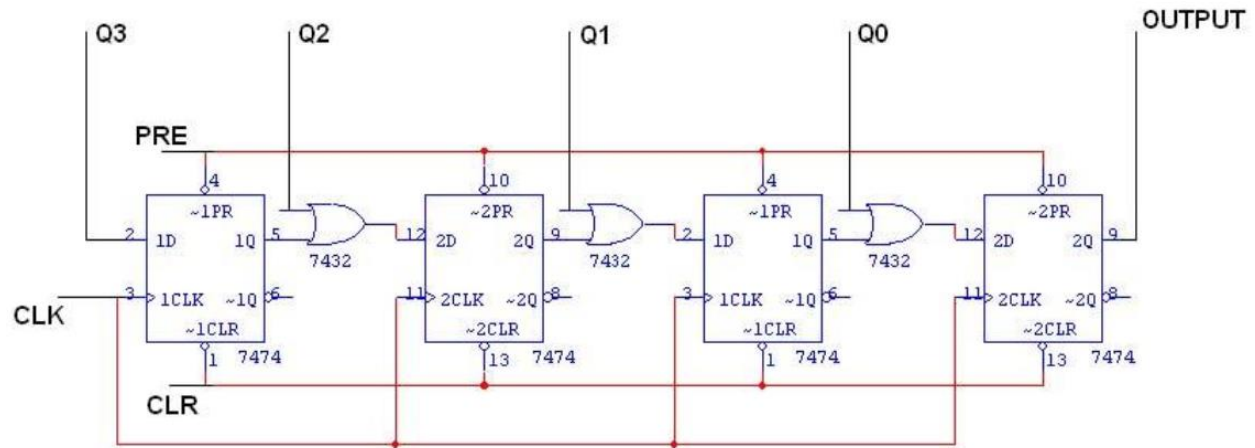
Serial-in to Parallel-out (SIPO) Shift Register:



Serial-in to Serial-out (SISO) Shift Register**Parallel-in to Serial-out (PISO) Shift Register****Parallel-in to Parallel-out (PIPO) Shift Register**

Serial-in to Serial-out (SISO) Shift Register:**Hardware Connection:****(01 Mark)****Truth Table:****(01 Mark)**

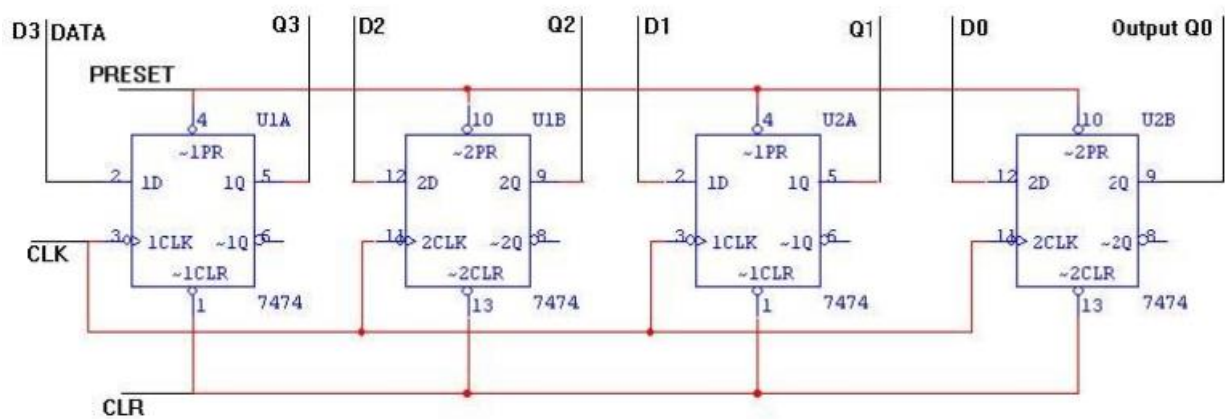
CLK	SERIAL IN	SERIAL OUT
1		
2		
3		
4		
5		
6		
7		

Parallel-in to Serial-out (PISO) Shift Register:**Hardware Connection:****(01 Mark)****Truth Table:****(01 Mark)**

CLK	Q3	Q2	Q1	Q0	OUTPUT
1					
2					
3					
4					
5					

Parallel-in to Parallel-out (PIPO) Shift Register

Hardware Connection: (01 Mark)



Truth Table: (01 Mark)

CLK	DATA INPUT				DATA OUTPUT			
	D1	D2	D3	D4	Q1	Q2	Q3	Q4
1								
2								

:

Observations:**(02 Mark)**

Review Questions

- 1) What is the difference between flip flop and shift register? How can you relate them? (01 Mark)

- 2) Why shift registers are an important part of sequential logic? (01 Mark)

- 3) Give some applications in which shift register can be used? (01 Mark)
