Adi Shankara

Institute of Engineering and Technology,
Vidya Bharathi Nagar, Kalady

Department of Computer Science and Engineering

Digital Laboratory Record

DEPARTMENT VISION

Nurturing globally competent Computer Science graduates capable of taking challenges in the Industry and Research & Development activities.

DEPARTMENT MISSION

M1: Imparting quality education to meet the needs of industry, and to achieve excellence in teaching and learning.
M2: Inculcating value-based, socially committed professionalism for development of society.
M3: Providing support to promote quality research.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: Graduate will be successfully employed in industry or academia
PEO2: Graduate shall be able to develop innovative computing solutions

PROGRAM OUTCOME (PO) & PROGRAM SPECIFIC OUTCOMES (PSO)

S No	PO
1	Engineering Knowledge
2	Problem Analysis
3	Design / Development of Solutions
4	Conduct Investigations of Complex Problems
5	Modern Tool Usage
6	The Engineer and society
7	Environment and Sustainability
8	Ethics
9	Individual and Team Work
10	Communication
11	Project Management and Finance
12	Life Long Learning
S No	**PSO**
1	Proficiency in Core Areas
2	Software Project Management

COURSE OUTCOMES

Student will be able to:

CO No.	Course Outcome	Knowledge Level
CO1	Design and implement combinational logic circuits using Logic Gates	Apply (K3)
CO2	Design and implement sequential logic circuits using Integrated Circuits	Apply (K3)
CO3	Simulate functioning of digital circuits using programs written in a Hardware	Apply (K3)
CO4	Function effectively as an individual and in a team to accomplish a given task of designing and implementing digital circuits	Apply (K3)

TABLE OF CONTENTS

PART - A

EXPERIMENT NO. 1

REALIZATION OF FUNCTIONS USING BASIC AND UNIVERSAL GATES

AIM OF THE EXPERIMENT:

a. To design and implement the given SOP function F (x2, x1, x0) = $\Sigma m(0, 1, 4, 5, 7)$. in AOI logic and NAND logic

b. Implement the given POS function F (x3, x2, x1, x0) = $\pi M(0, 1, 2,3,4,6)$ in AOI logic and NOR logic

LEARNING OBJECTIVE:

After completing this experiment, the student should be able to

- Realize the given SOP or POS expression using basic and universal gates

COMPONENTS REQUIRED:

Sl. No	Components	Specifications	Quantity
1.	IC 7400	Quad 2-input NAND gate	1
2.	IC 7402	Quad two-input NOR gate	1
3.	IC 7404	Hex inverter gate	1
4.	IC 7408	Quad 2-input AND gate	1
5.	IC 7432	Quad 2-input OR gate	1
6.	Resistor	330Ω	2
7.	LED		2

BRIEF DESCRIPTION:

Logic gates are connected together to produce a specified output for certain specified combinations of input variables, with no storage involved, the resulting circuits is called combinational logic. Logic minimization uses a variety of techniques to obtain the simplest gate level implementation of a logic function. Using Boolean laws, it is possible to minimize digital logic circuits. Since minimization with the use of Boolean laws is neither systematic nor suitable for computer implementation, a number of algorithms were proposed in order to overcome the implementation issue. Karnaugh proposed a technique for simplifying Boolean expressions using an elegant visual technique, which is actually a modified truth table intended to allow minimal sum-of products (SOP) and product- of-sums (POS) expressions to be obtained.

Any Boolean expression can be represented in the following forms

The Sum Of Products Form (SOP)

Each product term consists of the product of literals (variables or their complements). When two or more product terms are summed by Boolean addition, the resulting expression is SOP
Eg: AB+CD+ABC

The Standard SOP Form
A standard SOP expression is one in which all the variables in the domain appear in each product term (either in complemented or non-complemented form) in the expression
Eg: $ABC + A'BC + ABC'$

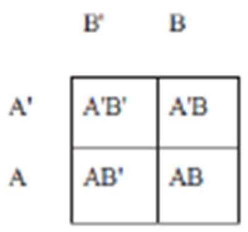
The Product of Sums (POS) form
A sum term consists of the Boolean addition of literals (variable or their complements). When two or more sum terms are multiplied, the resulting expression is a POS.
Eg: (A+B) (B+C+D)

The Standard POS forms
A standard POS expression is one in which all the variables in the domain appear in each sum term in the expression
Eg: (A+B+C) (A'+B'+C)

Boolean expressions and Truth tables
All standard Boolean expression can be easily converted into truth table format using binary values for each term in the expression.

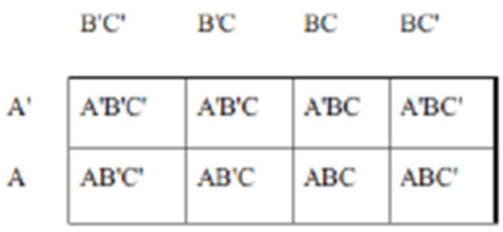
Karnaugh Map

The K map provides a systematic method for simplifying Boolean expressions and, if properly used, will produce the simplest SOP and POS expression possible, known as minimum expression. A Karnaugh map is similar to a truth table because it presents all of the possible values of input variables and the resulting output for each value.
The K-Map for 2,3 and 4 variable is shown in the following figure



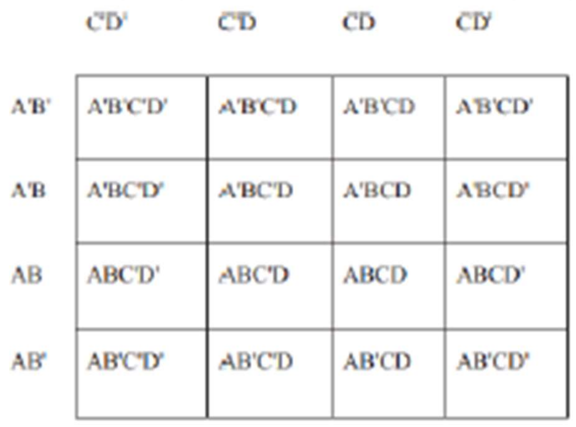
	B'	B
A'	A'B'	A'B
A	AB'	AB

2- Variable map
(4 Cells)



	B'C'	B'C	BC	BC'
A'	A'B'C'	A'B'C	A'BC	A'BC'
A	AB'C'	AB'C	ABC	ABC'

3- Variable map
(8 Cells)



	C'D'	C'D	CD	CD'
A'B'	A'B'C'D'	A'B'C'D	A'B'CD	A'B'CD'
A'B	A'BC'D'	A'BC'D	A'BCD	A'BCD'
AB	ABC'D'	ABC'D	ABCD	ABCD'
AB'	AB'C'D'	AB'C'D	AB'CD	AB'CD'

4- Variable

DESIGN:

Resistor Design:

Let,

I_D = LED forward current in Amps (found in the LED datasheet)

V_D = LED forward voltage drop in Volts (found in the LED datasheet)

VCC = supply voltage

R=(V_{CC}-V_D)/ID=(5-1.2)/10=380 Ω ≈ 330Ω

PROCEDURE:

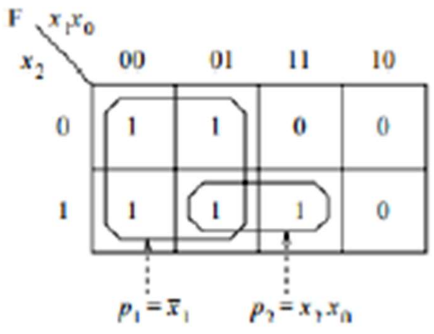
1. Wire the circuit as per the diagram on the bread board.
2. Apply various input combinations and observe output for each one.
3. Verify the truth table for each input/ output combination for function F1 and F2.

1. **Implement the given SOP function F(x2, x1, x0) = Σm(0, 1, 4, 5, 7). in AOI logic and NAND logic**

Table1:Truthtable of the given expression

INPUTS			OUTPUT
x 2	x1	x0	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

K map simplification



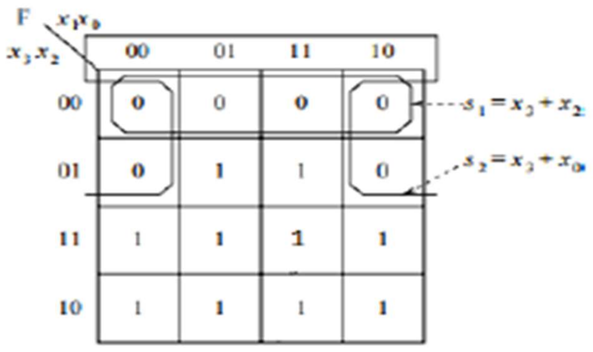
$$F = x1^{'} + x2\ x0$$

NAND
Logic

$$F = x1^{'} + x2\ x0$$

$$= \overline{\overline{x1^{'}} . \overline{(x2\ x0)}}$$

2. **Implement the given POS function F(x3, x2, x1, x0) = πM(0, 1, 2,3,4,6) in AOI logic and NOR logic**



$$F= (x3+x0)(x3+x2)$$

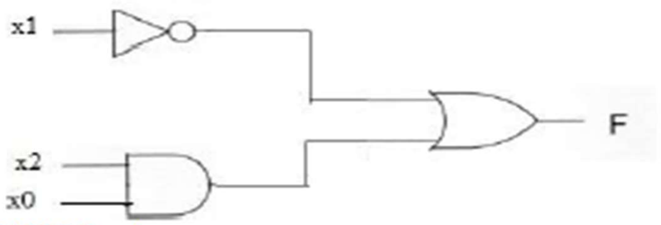
$$F= \overline{\overline{(x3+x0)(x3+x2)}}$$

$$= \overline{\overline{(x3+x0)}+\overline{(x3+x2)}}$$

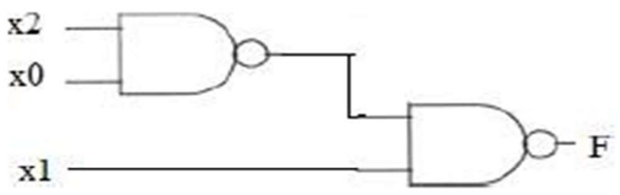
NOR logic

CIRCUIT DIAGRAM:

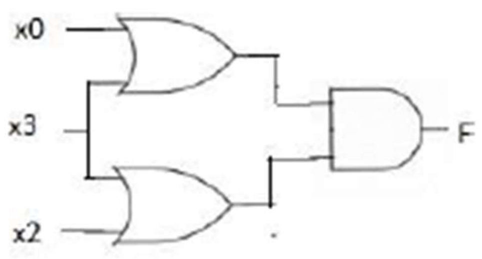
1. **AOI Logic**



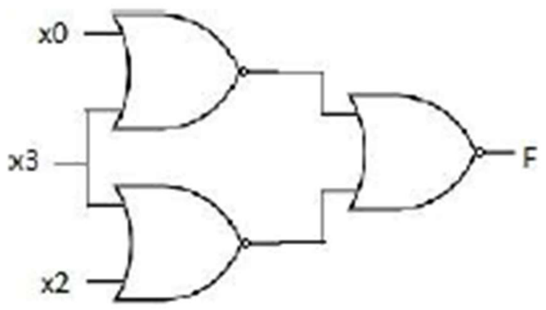
NAND Logic



2. **AOI Logic**



NOR Logic



INFERENCE:

Through this experiment, become familiar with two standard forms of logic functions: Sum of Products (SOP), and Product of Sums (POS). Studied the usage of K map for expression reduction. Studied the implementation of the reduced expression using AOI logic and Universal logic.

EXPERIMENT NO. 2a

HALF ADDER AND FULL ADDERS

AIM OF THE EXPERIMENT:

a) To design and set up a circuit that adds (i) 2 one-bit numbers (ii) 3 one-bit numbers using basic gates and universal gates.
b) To design and implement a Full-adder circuit using two half-adder circuits.

LEARNING OBJECTIVE:

After completing this experiment, the student will be able to:

- A half adder and full adder using basic gates and NAND only logic.
- Implement Full Adder circuits using two Half Adder

COMPONENTS REQUIRED:

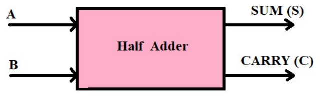
Sl. No	Components	Specifications	Quantity
1.	IC 7400	Quad 2-input NAND gate	3
2.	IC 7408	Quad 2-input AND gate	1
3.	IC 7432	Quad 2-input OR gate	2
4.	IC 7486	Quad 2-input X-OR gate	1
5.	LED		2
6.	Resistor	330Ω	2

BRIEF DESCRIPTION:

An adder is a digital logic circuit in electronics that implements addition of numbers. In many computers and other types of processors, adders are used to calculate addresses, similar operations and table indices in the ALU and also in other parts of the processors. Adders are classified into two types: half adder and full adder. The half adder circuit has two inputs: A and B, which add two inputs digits and generate a carry and sum. The full adder circuit has three inputs: A and C, which adds the three input numbers and generate a carry and sum. This article gives brief information about half adder and full adder in tabular forms and circuit diagrams.

Half Adder:

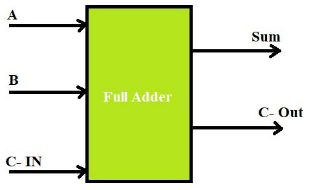
Half adders perform a simple binary addition of 2 bits producing 2 outputs, the sum bit (S) and carry bit (C).The half adder is shown in the block diagram in following figure



Block diagram for a Half Adder Circuit

Full Adder:

Full Adder is the adder which adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C-OUT and the normal output is designated as S which is SUM. Full adder logic is designed in such a manner that can take eight inputs together to create a byte-wide adder and cascade the carry bit from one adder to another. The full adder is shown in the block diagram in following figure



Block diagram for a Full Adder Circuit

DESIGN:

Resistor Design:

Let,

I_D = LED forward current in Amps (found in the LED datasheet)

V_D = LED forward voltage drop in Volts (found in the LED datasheet)

VCC = supply voltage

$R=(V_{CC}-V_D)/ID=(5-1.2)/10=380\ \Omega \approx 330\Omega$

HALF ADDER

Truth Table :

A (Input)	B (Input)	Sum S (Output)	Carry C (Output)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Using AOI Logic

S=A’B+AB’

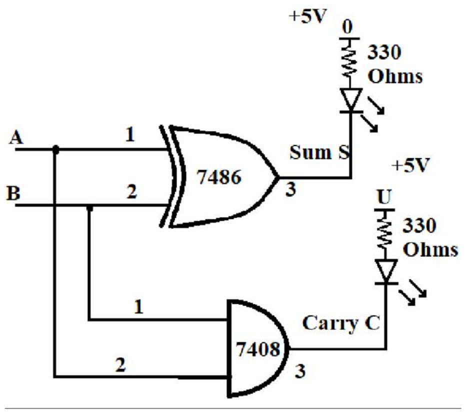
C=AB

Using NAND Logic

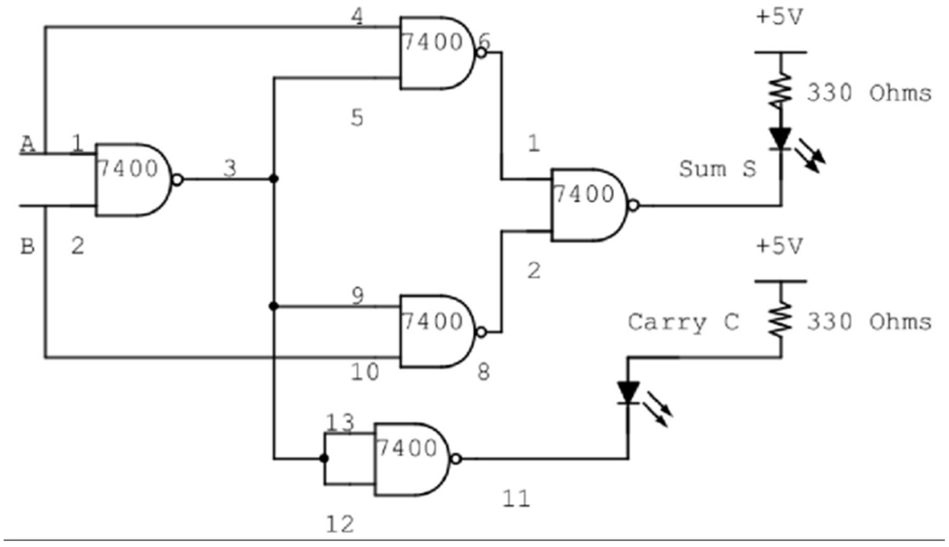
$$S = A\overline{B} + \overline{A}B = A\overline{B} + A\overline{A} + \overline{A}B + B\overline{B}$$
$$= A(\overline{A} + \overline{B}) + B(\overline{A} + \overline{B})$$
$$= A \cdot \overline{AB} + B \cdot \overline{AB}$$
$$= \overline{\overline{A \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}}}$$
$$C = AB = \overline{\overline{AB}}$$

CIRCUIT DIAGRAM:

Using AOI Logic



Using NAND Logic



FULL ADDER

Truth Table:

A (Input)	B (Input)	Cin (Input)	Sum S (Output)	Carry Cout (Output)
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

Using AOI Logic

$$S = \overline{A}\,\overline{B}C_{in} + \overline{A}B\overline{C}_{in} + A\overline{B}\,\overline{C}_{in} + ABC_{in} = A \oplus B \oplus C_{in}$$

and

$$C_{out} = \overline{A}BC_{in} + A\overline{B}C_{in} + AB\overline{C}_{in} + ABC_{in} = AB + (A \oplus B)C_{in}$$

Using NAND Logic

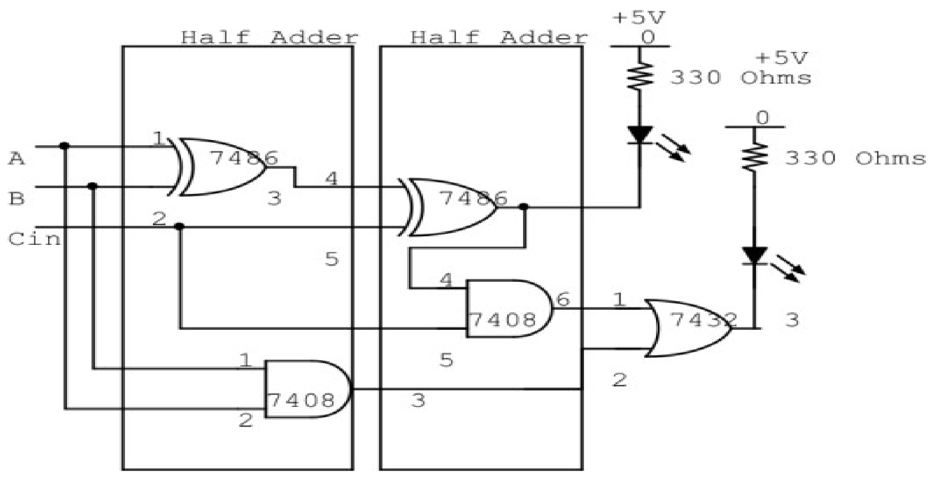
$$A \oplus B = \overline{\overline{A \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}}} = X.\ \text{Then}$$

$$S = A \oplus B \oplus C_{in} = \overline{\overline{X \cdot \overline{XC_{in}}} \cdot \overline{C_{in} \cdot \overline{XC_{in}}}} = X \oplus C_{in}$$

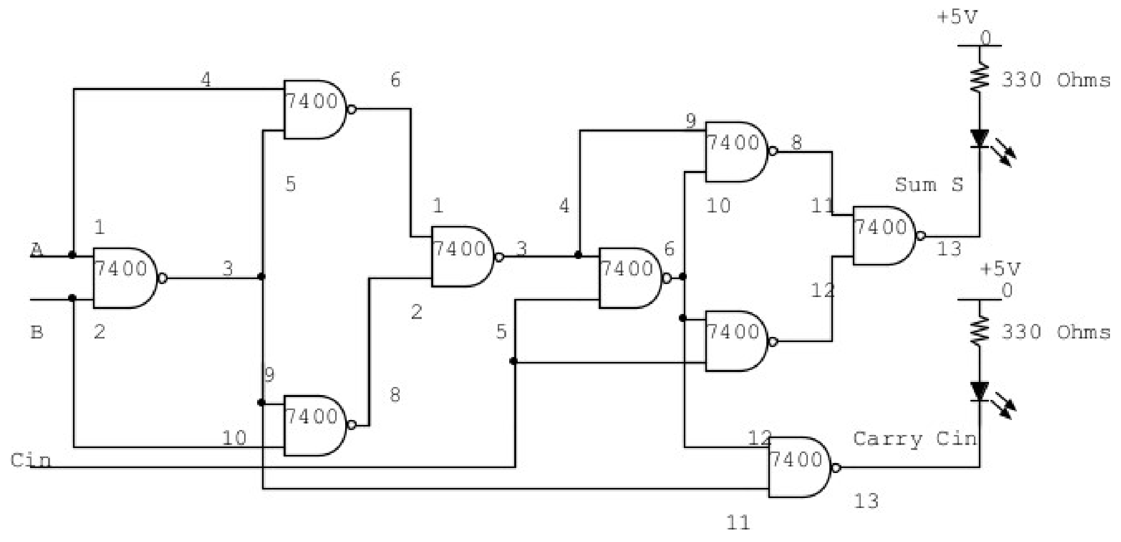
$$C_{out} = C_{in}(A \oplus B) + AB = \overline{\overline{C_{in}(A \oplus B)} \cdot \overline{AB}}$$

CIRCUIT DIAGRAM:

Using AOI Logic



Using NAND Logic



PROCEDURE:

1. Wire the circuit as per the diagram on the bread board.
2. Apply various input combinations and observe output for each one.
3. Verify the truth table for each input/ output combination.

INFERENCE:

Half adder circuit can be used for adding two one bit numbers. Two half adder can be used for implementing a full adder which performs addition of three one bit numbers.

EXPERIMENT NO. 2b

HALF SUBTRACTOR AND FULL SUBTRACTOR

AIM OF THE EXPERIMENT:

c) To design and set up a circuit that subtract (i) 2 one-bit numbers (ii) 3 one-bit numbers using basic gates and universal gates.

LEARNING OBJECTIVE:

After completing this experiment, the student will be able to:

- A half subtractor and full subtractor using basic gates and NOR only logic.

COMPONENTS REQUIRED:

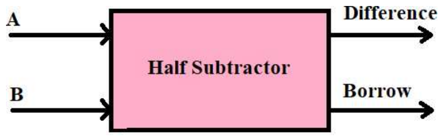
Sl. No	Components	Specifications	Quantity
1.	IC 7408	Quad 2-input AND gate	1
2.	IC 7432	Quad 2-input OR gate	1
3.	IC 7402	Quad 2-input NOR gate	3
4.	IC 7486	Quad 2-input X-OR gate	1
5.	IC 7404	Hex inverter gate	1
6.	LED		2
7.	Resistor	330Ω	2

BRIEF DESCRIPTION:

A subtractor is a digital logic circuit in electronics that implements the subtraction of numbers. In many computers and other types of processors, subtractors are used to calculate addresses, similar operations and table indices in the ALU and also in other parts of the processors. Subtractors are classified into two types: half subtractor and full subtractor. The half subtractor circuit has two inputs: A and B, which subtract two inputs digits and generate a carry and sum. The full subtractor circuit has three inputs: A and C, which subtracts the three input numbers and generate a carry and sum. This article gives brief information about hsubtractorsctor and full subtractor in tabular forms and circuit diagrams.

Half Subtractor:

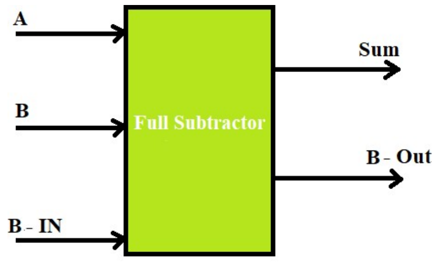
Half subtractors perform a simple binary subtraction of 2 bits producing 2 outputs, the sum bit (S) and carry bit (C).The half subtractor is shown in the block diagram in following figure



Block diagram for a Half Subtractor Circuit

Full Subtractor:

Full Subtractor is the subtractor which subtracts three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C-OUT and the normal output is designated as S which is SUM. Full subtractor logic is designed in such a manner that can take eight inputs together to create a byte-wide subtractor and cascade the carry bit from one subtractor to another. The full subtractor is shown in the block diagram in following figure.



Block diagram for a Full Subtractor Circuit

DESIGN:

<u>Resistor Design:</u>

Let,

I_D = LED forward current in Amps (found in the LED datasheet)

V_D = LED forward voltage drop in Volts (found in the LED datasheet)

VCC = supply voltage

R=(VCC-VD)/ID=(5-1.2)/10=380 Ω ≈ 330 Ω

HALF SUBTRACTOR

Truth Table :

A	B	D	B_o
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Using AOI Logic

d=A’B+AB’

b=A’B

Using NOR Logic

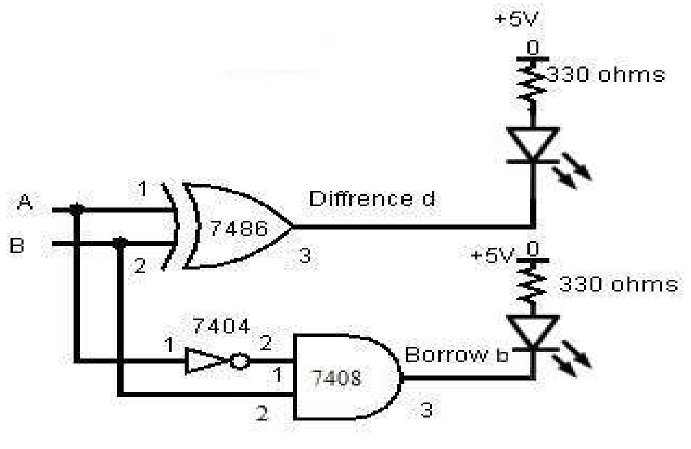
$$d = A \oplus B = A\overline{B} + \overline{A}B = A\overline{B} + B\overline{B} + \overline{A}B + A\overline{A}$$

$$= \overline{B}(A + B) + \overline{A}(A + B) = \overline{B + \overline{A + B}} + \overline{A + \overline{A + B}}$$

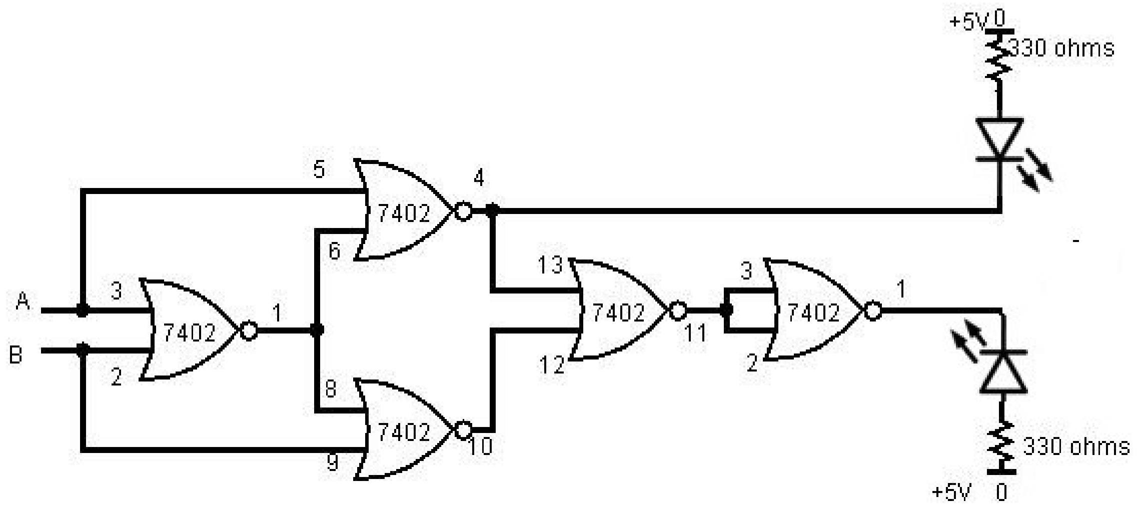
$$d = \overline{A}B = \overline{A}(A + B) = \overline{\overline{\overline{A}(A + B)}} = \overline{A + \overline{(A + B)}}$$

CIRCUIT DIAGRAM :

Using AOI Logic



Using NOR Logic



FULL SUBTRACTOR

Truth Table:

A	B	B_{in}	D	B_o
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

Using AOI Logic

$$\begin{aligned} d &= \bar{A}\bar{B}b_i + \bar{A}B\bar{b}_i + A\bar{B}\bar{b}_i + ABb_i \\ &= b_i(AB + \bar{A}\bar{B}) + \bar{b}_i(A\bar{B} + \bar{A}B) \\ &= b_i(\overline{A \oplus B}) + \bar{b}_i(A \oplus B) = A \oplus B \oplus b_i \end{aligned}$$

and

$$\begin{aligned} b &= \bar{A}\bar{B}b_i + \bar{A}B\bar{b}_i + \bar{A}Bb_i + ABb_i = \bar{A}B(b_i + \bar{b}_i) + (AB + \bar{A}\bar{B})b_i \\ &= \bar{A}B + (\overline{A \oplus B})b_i \end{aligned}$$

Using NOR Logic

$$\begin{aligned} d &= A \oplus B \oplus b_i = \overline{\overline{(A \oplus B) \oplus b_i}} \\ &= \overline{(A \oplus B)b_i + (\overline{A \oplus B})\bar{b}_i} \\ &= \overline{[(A \oplus B) + (\overline{A \oplus B})\bar{b}_i][b_i + (\overline{A \oplus B})\bar{b}_i]} \end{aligned}$$

$$= \overline{(A \oplus B) + \overline{(A \oplus B) + b_i}} + \overline{b_i + \overline{(A \oplus B) + b_i}}$$

$$= \overline{\overline{\overline{(A \oplus B) + \overline{(A \oplus B) + b_i}} + \overline{b_i + \overline{(A \oplus B) + b_i}}}}$$

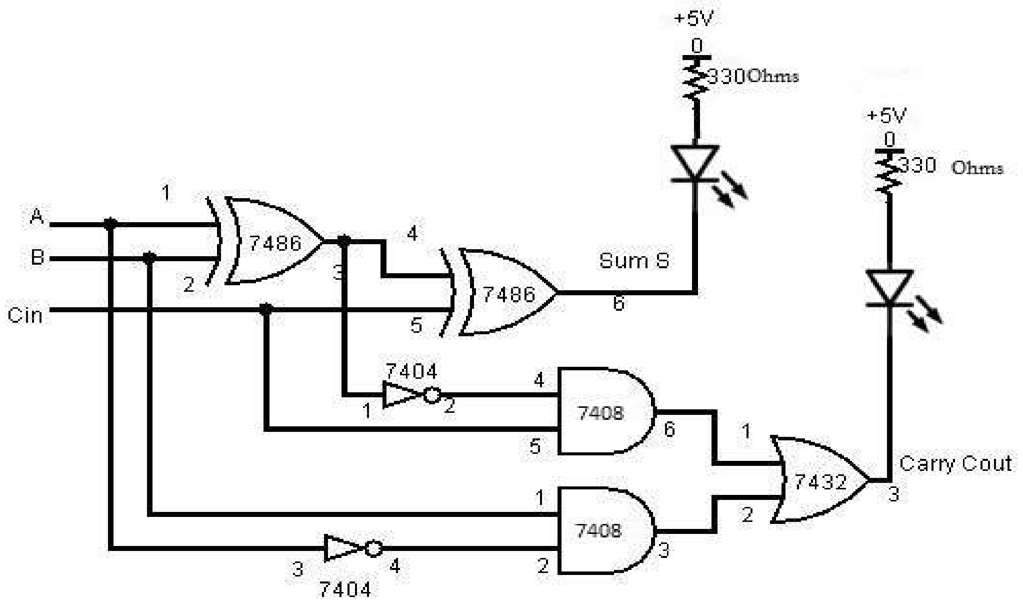
$$b = \bar{A}B + b_i(\overline{A \oplus B})$$

$$= \bar{A}(A + B) + (\overline{A \oplus B})[(A \oplus B) + b_i]$$

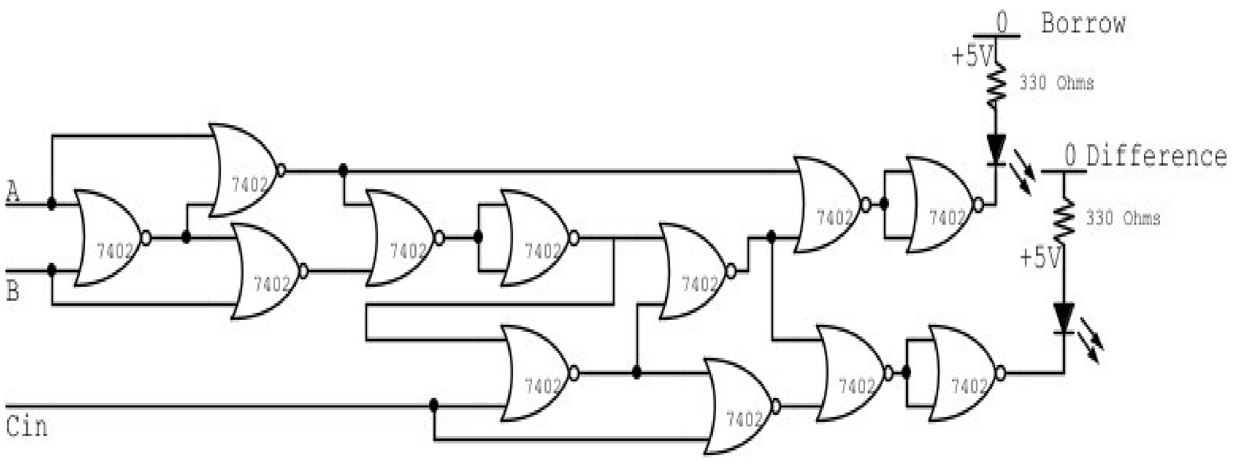
$$= \overline{\overline{\overline{A + \overline{A + B}} + \overline{(A \oplus B) + \overline{(A \oplus B) + b_i}}}}$$

CIRCUIT DIAGRAM:

Using AOI Logic



Using NOR Logic



PROCEDURE:

1. Wire the circuit as per the diagram on the bread board.
2. Apply various input combinations and observe output for each one.
3. Verify the truth table for each input/ output combination.

INFERENCE:

Half subtractor circuit can be used for subtracting two one bit numbers. Two half subtractor can be used for implementing a full subtractor which performs subtraction of three one bit numbers

EXPERIMENT NO. 3

CODE CONVERTERS

AIM OF THE EXPERIMENT:

Design and implement BCD to Excess 3 and Binary to Gray code converters.

LEARNING OBJECTIVE:

After completing this experiment, the student will be able to:

- Implement BCD to Excess 3 code converter using basic gates
- Implement Binary to Gray code converter using basic gates

COMPONENTS REQUIRED:

Sl. No	Components	Specifications	Quantity
1.	IC 7408	Quad 2-input AND gate	1
2.	IC 7432	Quad 2-input OR gate	1
3.	IC 7404	Hex inverter gate	1
4.	IC 7486	Quad 2-input X-OR gate	1
5.	LED		8
6.	Resistor	330Ω	8

BRIEF DESCRIPTION:

Binary coded decimal (BCD) is a system of writing numerals that assigns a four-digit binary code to each digit 0 through 9 in a decimal (base-10) numeral. The four-bit BCD code for any particular single base-10 digit is its representation in binary notation, Numbers larger than 9, having two or more digits in the decimal system, are expressed digit by digit.

The excess-3 code is a non-weighted code used to express code used to express decimal numbers. It is a self-complementary binary coded decimal (BCD) code and numerical system which has biased representation. It is particularly significant for arithmetic operations as it overcomes shortcoming encountered while using 8421 BCD code to add two decimal digits whose sum exceeds 9. Excess-3 arithmetic uses different algorithm than normal non-biased BCDor binary positional number system.

Binary number system is a number system expressed in the base-2 numeral system or binary numeral system, a method of mathematical expression which uses only two symbols: typically "0" and "1". The base-2 numeral system is a positional notation with a radix of 2. Each digit is referred to as a bit, or binary digit

Gray Code is the minimum-change code category of coding in which, the two consecutive values changes by only a single bit. More specifically we can say, it is a binary number system where while moving from one step to the next, only a single bit shows variation. It is an unweighted code, as here like other number systems, no particular weight is provided to various bit positions.

DESIGN:

<u>Resistor Design:</u>

Let,

I_D	= LED forward current in Amps (found in the LED datasheet)
V_D	= LED forward voltage drop in Volts (found in the LED datasheet)
VCC	= supply voltage

R=(V_{CC}-V_D)/ID=(5-1.2)/10=380 Ω ≈ 330Ω

<u>BCD to Excess 3 code converter</u>

A BCD digit can be converted to its corresponding Excess-3 code by simply adding 3 to it.

Let A, B, C and D be the bits representing the binary numbers, where D is the LSB and A is the MSB, and

Let w, x, y and z be the bits representing the gray code of the binary numbers, where z is the LSB and w is the MSB.

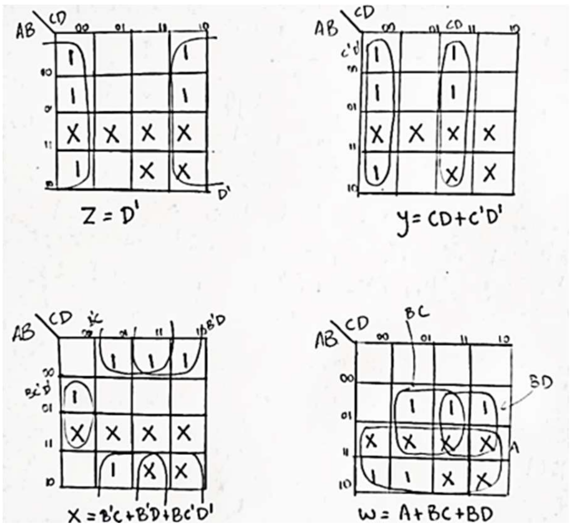
Truth Table:

BCD(8421)				Excess-3			
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

The X’s mark don’t care conditions.

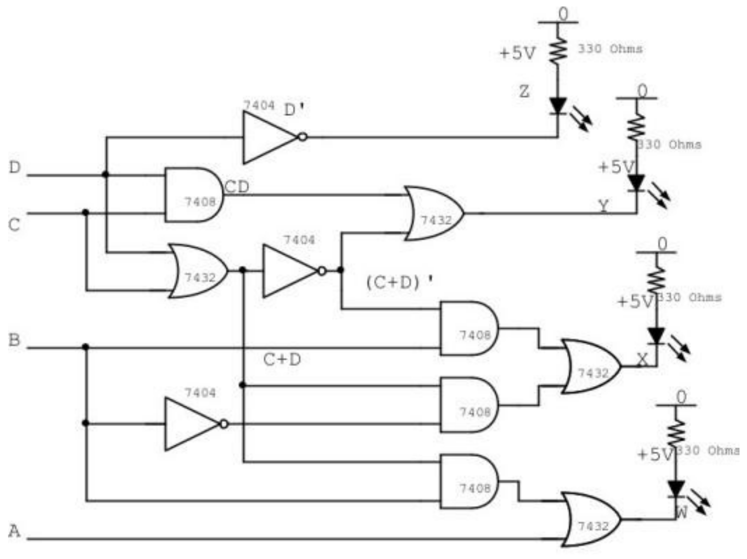
To find the corresponding digital circuit, K-Map technique is used for each of the Excess-3 code bits as output with all of the bits of the BCD number as input.

Using K-map method to design the logical circuit for the conversion of BCD to Excess-3 code:



w=A+BC+BD x=B' C+B' D+BC' D' y=CD+C'D' z=D'

CIRCUIT DIAGRAM:



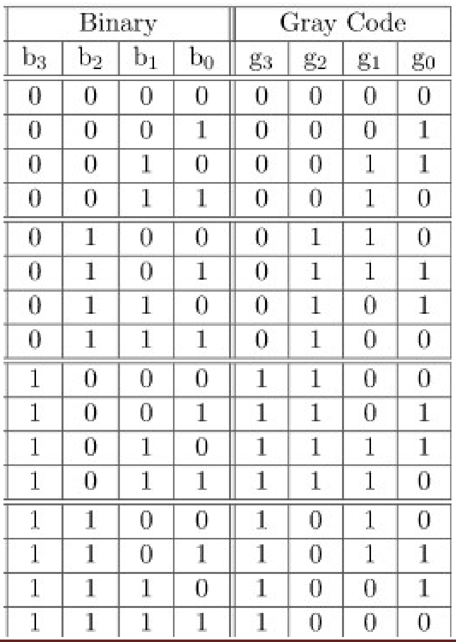
BCD to Excess 3 code converter

<u>**Binary to Gray code converter**</u>

By putting the MSB of 1 below the axis and the MSB of 1 above the axis and reflecting the (n-1) bit code about an axis after 2^{n-1} rows, we can obtain the n-bit gray code.

Let b_0, b_1, b_2 and b_3 be the bits representing the binary numbers, where b_0 is the LSB and b_3 is the MSB, and
Let g_0, g_1, g_2 and g_3 be the bits representing the gray code of the binary numbers, where $g_0$is the LSB and g_3 is the MSB.

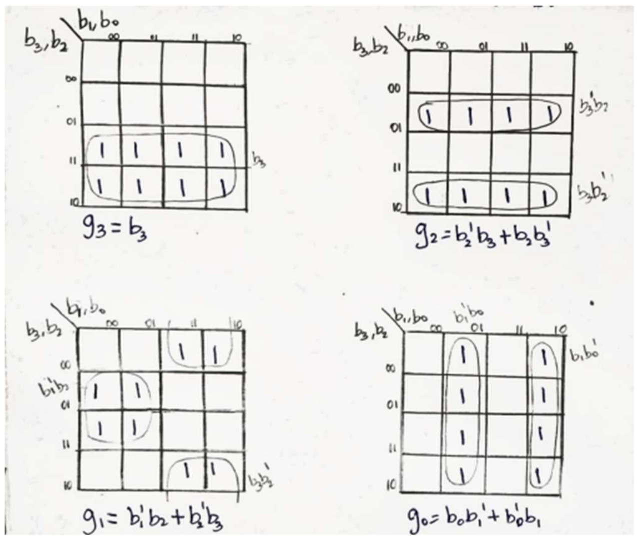
Truth Table:



Binary				Gray Code			
b_3	b_2	b_1	b_0	g_3	g_2	g_1	g_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

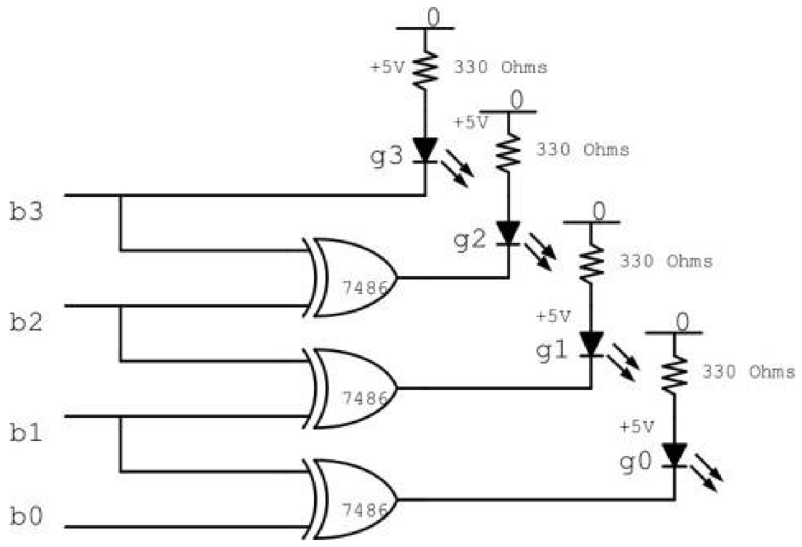
To find the corresponding digital circuit, K-Map technique is used for each of the gray code bits as output with all of the bits of the binary as input.

Using K-map method to design the logical circuit for the conversion of binary to gray code:



g3= b3 g2= b2b3’+b3b2’=b2⨁b3 g1=b2b1’+b1b2’=b1⨁b2 g0=b0b1’+b1b0’=b0⨁b1

CIRCUIT DIAGRAM:



PROCEDURE:

1. Wire the circuit as per the diagram on the bread board.
2. Apply various input combinations and observe output for each one.
3. Verify the truth table for each input/ output combination.

INFERENCE:

GCD to Excess 3 convertor can be used for convert BCD code to Excess 3 code and Binary to Gray code converter can be used for convert Binary to Gray code. Both the converters were implemented using basic gates.

EXPERIMENT NO. 4

IMPLEMENTATION OF FLIP FLOPS

AIM OF THE EXPERIMENT:

To study the working of S-R, J-K, T& D flip-flops, implemented using NAND gates, and to verify their truth tables

LEARNING OBJECTIVE:

Upon completion of this experiment, students will be able to:

● Understand different types of flip-flops, and their truth tables

● Construct different types of flip-flops using the universal NAND gate and therebyverify their truth tables

COMPONENTS REQUIRED:

Sl. No	Components	Specifications	Quantity
1.	IC 7400	Quad 2-input NAND gate	3
2.	IC 7404	Hex inverter gate	1
3.	LED		2
4.	Resistor	330Ω	2

BRIEF DESCRIPTION:

A flip flop is an electronic circuit with two stable states that can be used to store binary data. The stored data can be changed by applying varying inputs. Flip-flops and latches are fundamental building blocks of digital electronics systems used in computers, communications, and many other types of systems. Flip-flops and latches are used in computers, communications,and many other types of systems. Flip-flops and latches are used as data storage elements. It is the basic storage element in sequential logic. The basic difference between a latch and a flip-flopis a gating or clocking mechanism. A flip flop is synchronous and is also known as gated or clocked SR latch. There are mainly 4 types of flip-flops: SR, JK, T & D. The SR flip-flop is the implest and easiest to understand. However, if both S & R inputs are given 1, the flip-flop moves into invalid state.

Due to the undefined state in the SR flip flop, another flip-flop is required in electronics. The JK flip flop is an improvement on the SR flip flop where S=R=1 is not a problem. The input condition of J=K=1, gives an output that is an inversion of the previousstate. D flip flop is a better alternative that is very popular with digital electronics. They are commonly used for counters and shift-registers and input synchronization. The output Q is sameas the input. A T flip flop is like the JK flip-flop. These are basically single input version of JK flip flops. This modified form of JK flip-flop is obtained by connecting both inputs J and K together. This flip-flop has only one input along with the clock input. These flip-flops are called T flip-flops because of their ability to complement their state (i.e) Toggle. The applications of Flip-flops are specified below.

- Counters
- Frequency Dividers
- Shift Registers
- Storage Registers

DESIGN:

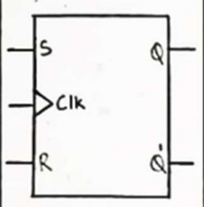
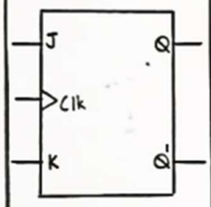
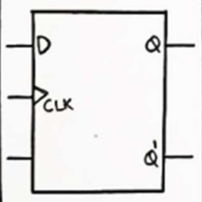
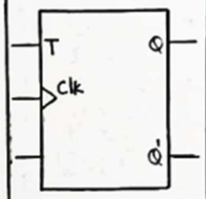
<u>Resistor Design:</u>

Let,

I_D	= LED forward current in Amps (found in the LED datasheet)
V_D	= LED forward voltage drop in Volts (found in the LED datasheet)
VCC	= supply voltage

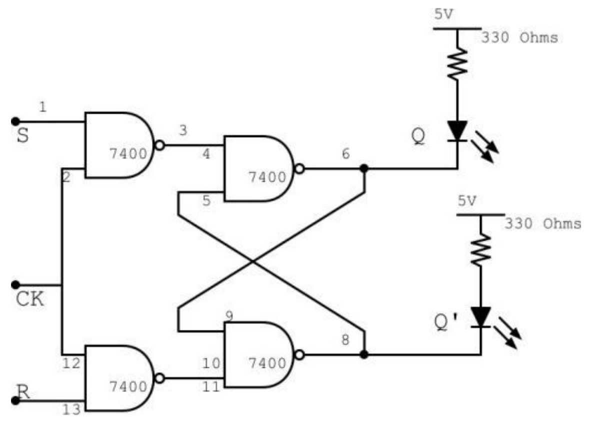
$R=(V_{CC}-V_D)/ID= (5-1.2)/10=380\ \Omega \approx 330\Omega$

Characteristics of Flip Flops:

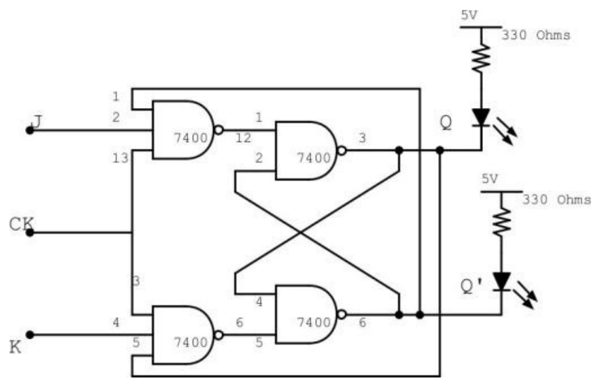
Flip-Flop Name	Flip-Flop symbol	Characteristic Table	Characteristic Equation	Excitation Table
SR		S R Q(next) 0 0 Q 0 1 0 1 0 1 1 1 ?	$Q(next) = S + R'Q$ $SR = 0$	Q Q(next) S R 0 0 0 X 0 1 1 0 1 0 0 1 1 1 X 0
JK		J K Q(next) 0 0 Q 0 1 0 1 0 1 1 1 Q'	$Q(next) = JQ' + K'Q$	Q Q(next) J K 0 0 0 X 0 1 1 X 1 0 X 1 1 1 X 0
D		D Q(next) 0 0 1 1	$Q(next) = D$	Q Q(next) D 0 0 0 0 1 1 1 0 0 1 1 1
T		T Q(next) 0 Q 1 Q'	$Q(next) = TQ' + T'Q$	Q Q(next) T 0 0 0 0 1 1 1 0 1 1 1 0

CIRCUIT DIAGRAM:

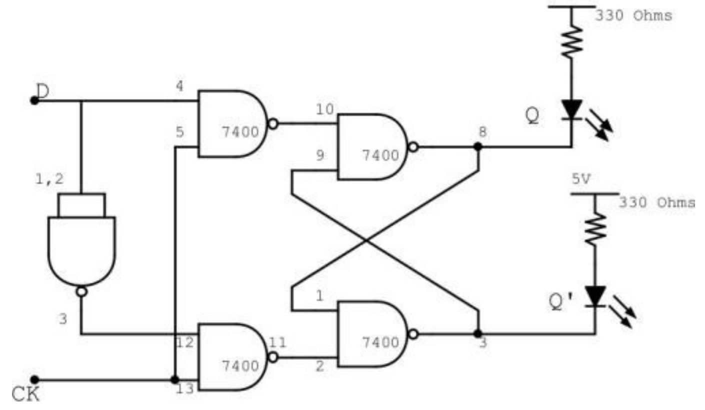
S-R flip-flop using NAND gates:



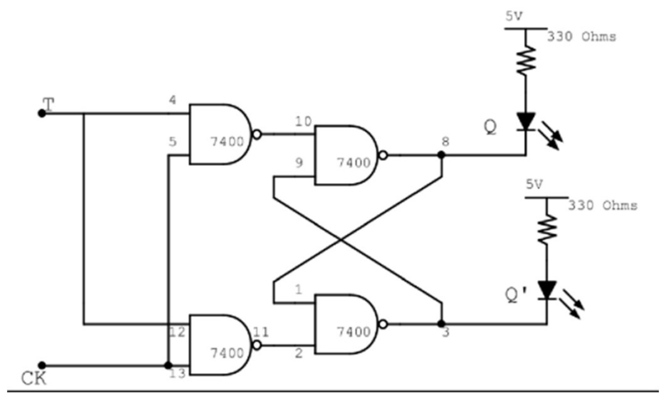
J-K flip-flop using NAND gates:



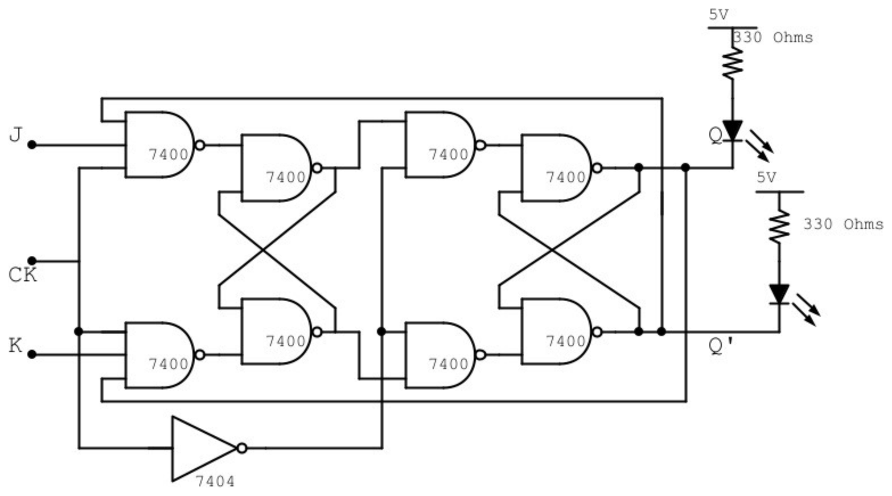
D flip-flop using NAND gates:



T flip-flop using NAND gates:



Master-Slave using NAND gates:



PROCEDURE:

1. Wire the circuit as per the diagram on the bread board.
2. Apply various input combinations and observe output for each one.
3. Verify the truth table for each input/ output combination.
4. Repeat the process for all flip-flop.

INFERENCE:

Flip-flops SR, JK, T & D were realized using universal gate, NAND and their truth tables were verified.

EXPERIMENT NO. 5

ASYNCHRONOUS COUNTER: DESIGN AND IMPLEMENT 3 BIT UP / DOWN COUNTER

AIM OF THE EXPERIMENT:

To study 3 bit asynchronous Up / Down counter using flip flops.

LEARNING OBJECTIVE:

Upon completion of this experiment, students will be able to:

- Understand the concept of Counters.
- Implement 3 bit asynchronous Up / Down counter using flip flops

COMPONENTS REQUIRED:

Sl. No	Components	Specifications	Quantity
1.	IC 7404	Hex Inverter	1
2.	IC 7408	Quad AND gate	1
3.	IC 7432	Quad OR gate	1
4.	IC 7473	Dual J-K Flip-Flop	2
5.	LED		3
6.	Resistor	330Ω	3

BRIEF DESCRIPTION:

A counter is a register capable of counting number of clock pulse arriving at its clock input. Counter represents the number of clock pulses arrived. A specified sequence of states appears as counter output. This is the main difference between a register and a counter. There are two types of counters, synchronous and asynchronous. In synchronous common clock is given to all flip flop and in asynchronous first flip flop is clocked by external pulse and then each successive flip flop is clocked by Q or Q output of previous stage. Soon the clock ofsecond stage is triggered by output of first stage. Because of inherent propagation delay time all flip flops are not activated at same time which results in asynchronous operation.

DESIGN:

<u>Resistor Design:</u>

Let,

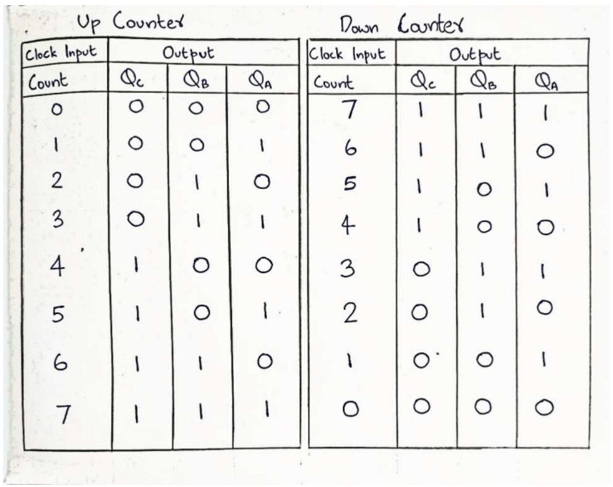
I_D = LED forward current in Amps (found in the LED datasheet)

V_D = LED forward voltage drop in Volts (found in the LED datasheet)

VCC = supply voltage

$R=(V_{CC}-V_D)/ID= (5-1.2)/10=380\ \Omega \approx 330\Omega$

TRUTH TABLE:



Up Counter

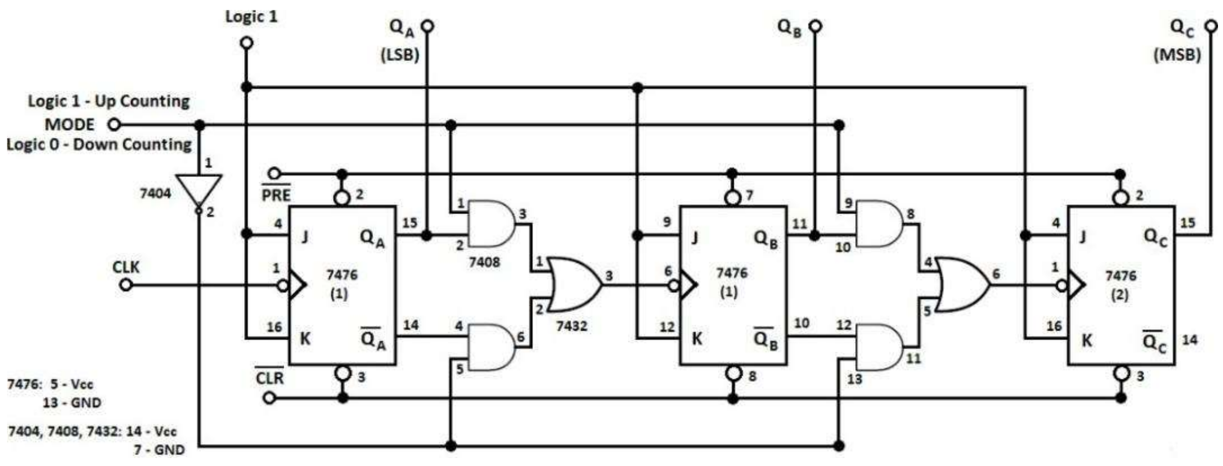
Clock Input	Output		
Count	Q_C	Q_B	Q_A
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Down Counter

Clock Input	Output		
Count	Q_C	Q_B	Q_A
7	1	1	1
6	1	1	0
5	1	0	1
4	1	0	0
3	0	1	1
2	0	1	0
1	0	0	1
0	0	0	0

CIRCUIT DIAGRAM:

3 bit asynchronous Up / Down counter



PROCEDURE:

1. Wire the circuit as per the diagram on the bread board.
2. Apply various input combinations and observe output for each one.
3. Verify the truth table for each input/ output combination.

INFERENCE:

3 bit asynchronous Up / Down counter were realized using J-K Flip-Flops and their truth tables were verified.

EXPERIMENT NO. 6

ASYNCHRONOUS COUNTER: REALIZATION OF MOD N COUNTERS

AIM OF THE EXPERIMENT:

Realization of 3-bit Asynchronous Mod-N counter design.

LEARNING OBJECTIVE:

Upon completion of this experiment, students will be able to:

- Understand the concept of Asynchronous Counters.
- Implement 3-bit asynchronous Mod-5 counter using flip flops

COMPONENTS REQUIRED:

Sl. No	Components	Specifications	Quantity
1.	IC 7473	Dual J-K Flip-Flop	2
2.	IC 7400	Quad NAND gate	1
3.	LED		3
4.	Resistor	330Ω	3

BRIEF DESCRIPTION:

Asynchronous counters are those whose output is free from the clock signal. Because the flip flops in asynchronous counters are supplied with different clock signals, there may be delay in producing output.The required number of logic gates to design asynchronous counters is very less. So they are simple in design. Another name for Asynchronous counters is “Ripple counters”.The number of flip flops used in a ripple counter is depends up on the number of states of counter (ex: Mod 4, Mod 2 etc). The number of output states of counter is called “Modulus” or“MOD” of the counter. The maximum number of states that a counter can have is 2n where n represents the number of flip flops used in counter.

DESIGN:

Resistor Design:

Let,

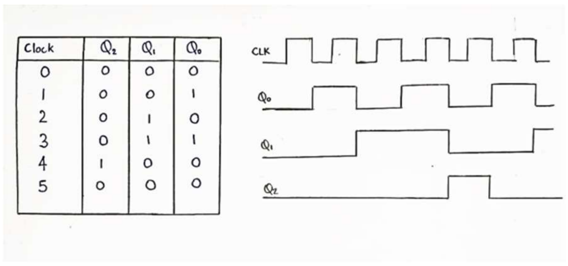
I_D = LED forward current in Amps (found in the LED datasheet)

V_D = LED forward voltage drop in Volts (found in the LED datasheet)

VCC = supply voltage

$R=(V_{CC}-V_D)/ID= (5-1.2)/10=380\ \Omega \approx 330\Omega$

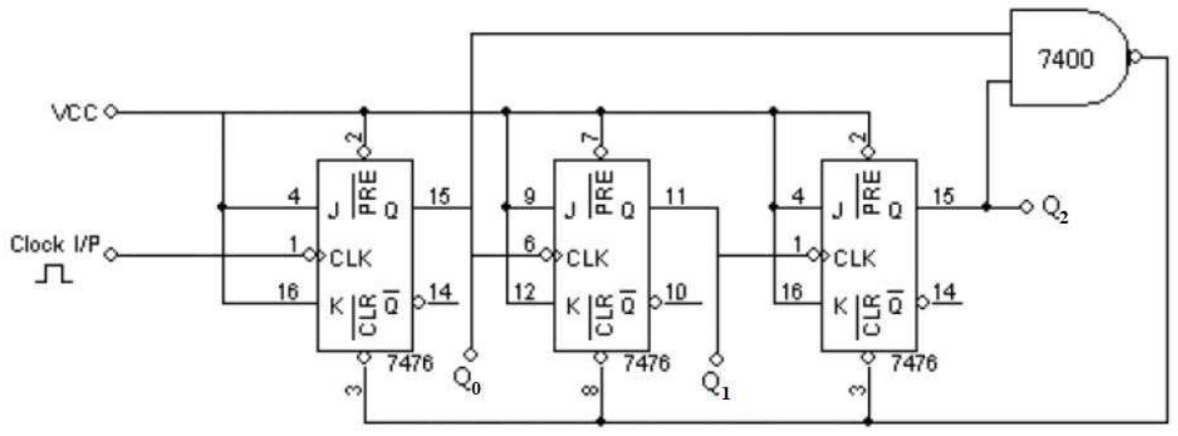
TRUTH TABLE:



Clock	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	0	0	0

CIRCUIT DIAGRAM:

3 bit Mod-5 Asynchronous counter



PROCEDURE:

1. Wire the circuit as per the diagram on the bread board.
2. Observe output for each one.
3. Verify the truth table for each input/ output combination.

INFERENCE:

3-bit Asynchronous Mod-N counter was realized using J-K Flip-Flops and their truth tables were verified.

EXPERIMENT NO. 7a

MOD-5 SYNCHRONOUS COUNTERS

AIM OF THE EXPERIMENT:

To realize a synchronous mod-5 Counter using IC 7473.

LEARNING OBJECTIVE:

Upon completion of this experiment, students will be able to:

- Understand the construction of mod-N counters.

COMPONENTS REQUIRED:

Sl. No	Components	Specifications	Quantity
1.	IC 7473	Dual J-K Flip-Flop with asynchronous clear	2
2.	IC 7408	Quad 2 input AND gate	1
3.	LED		3
4.	Resistor	330Ω	3

BRIEF DESCRIPTION:

Counters which advance their sequence of numbers or states when activated by a clock input are said to operate in a "count up" mode. Likewise, counters which decrease their sequenceof numbers or states when activated by a clock input are said to operate in a "count down" mode.Counters that operate in both the UP and DOWN modes are called bidirectional counters.

Counters are sequential logic devices that are activated or triggered by an external timing pulse or clock signal. A counter can be constructed to operate as a synchronous circuit or an asynchronous circuit. With synchronous counters, all the data bits change synchronously with theapplication of a clock signal. Whereas an asynchronous counter circuit is independent of the input clock so the data bits change state at different times one after the other. Then counters are sequential logic devices that follow a predetermined sequence of counting states which are triggered by an external clock (CLK) signal. The number of states or counting sequences through which a particular counter advances before returning once again back to its original first state is called the modulus (MOD). In other words, the modulus (or just modulo) is the number of states the counter counts and is dividing number of the counter. Modulus counters, or simply MOD counters are defined based on the number of states that the counter will sequence through before returning back to its original value. For example: a 2-bit counter that counts from 002 to 112 in binary, that is 0 to 3 in decimal, has a modulus value of 4 (00→01→10→11, returns back to 00) so would therefore be called a modulo-4, or mod-4 counter. Note also that it has taken 4 clock pulses to get from 00 to 11. As in this simple example there are only two bits, (n= 2) then the maximum number of possible output states (maximum modulus) for the counter is $2^n = 2^2$ or 4. However, counters can be designed to count to any number of 2n states in their sequence by cascading together multiple counting stages to produce a single modulus or MOD-N counter. Therefore, a “Mod-N” counter will require “N” number of flip-flops connected together to count a single data bit while providing 2^n different output states, (n is the number of bits). Note that N is always a whole integer value. Then we can see that MOD counters have a modulus value that is an integral power of 2, that is 2, 4, 8, 16 and so on to produce an n-bit counter depending on the number of flip-flops used, and how they are connected, determining the type and modulus ofthe counter.

DESIGN:

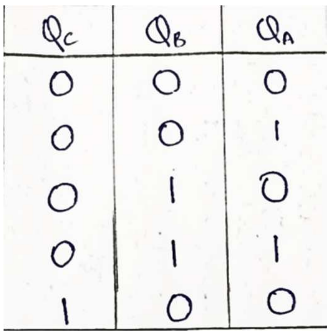
Resistor Design:

Let,

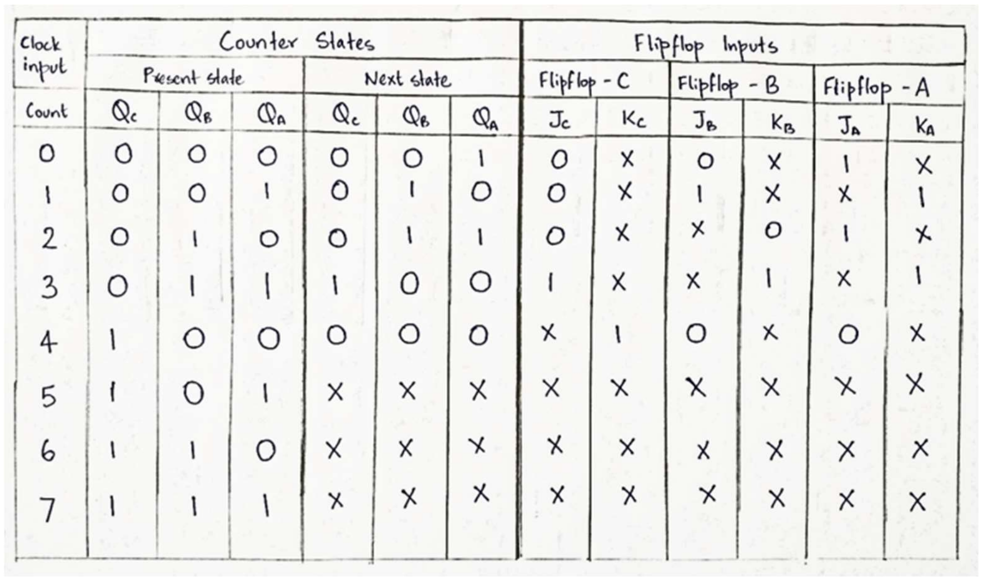
I_D	= LED forward current in Amps (found in the LED datasheet)
V_D	= LED forward voltage drop in Volts (found in the LED datasheet)
V_{CC}	= supply voltage

$R = (V_{CC}-V_D)/ID = (5-1.2)/10 = 380\ \Omega \approx 330\Omega$

TRUTH TABLE:



Q_C	Q_B	Q_A
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0

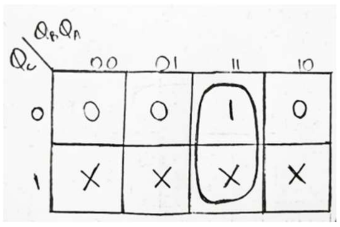


Clock input	Counter States						Flipflop Inputs					
	Present state			Next state			Flipflop - C		Flipflop - B		Flipflop - A	
Count	Q_C	Q_B	Q_A	Q_C	Q_B	Q_A	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	0	1	0	X	0	X	1	X
1	0	0	1	0	1	0	0	X	1	X	X	1
2	0	1	0	0	1	1	0	X	X	0	1	X
3	0	1	1	1	0	0	1	X	X	1	X	1
4	1	0	0	0	0	0	X	1	0	X	0	X
5	1	0	1	X	X	X	X	X	X	X	X	X
6	1	1	0	X	X	X	X	X	X	X	X	X
7	1	1	1	X	X	X	X	X	X	X	X	X

$2^n > 5$

n-3, is the number of flipflops are required.

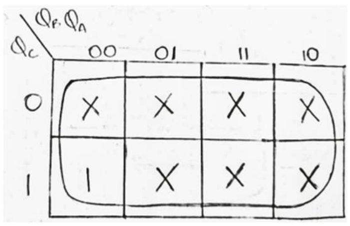
J_C:



$Q_C \backslash Q_BQ_A$	00	01	11	10
0	0	0	1	0
1	X	X	X	X

$J_C = Q_BQ_A$

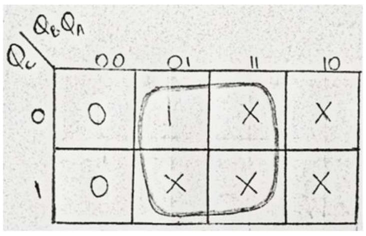
K_C:



$Q_C \backslash Q_BQ_A$	00	01	11	10
0	X	X	X	X
1	1	X	X	X

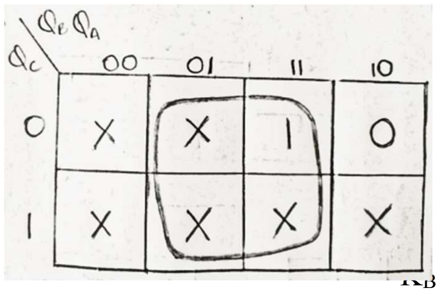
$K_C = 1$

J_B:



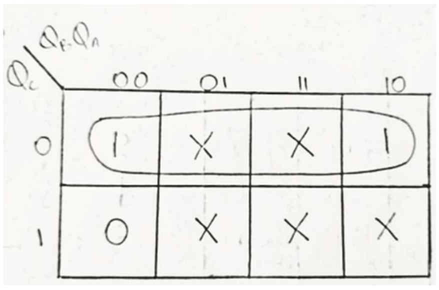
$J_B = Q_A$

K_B:



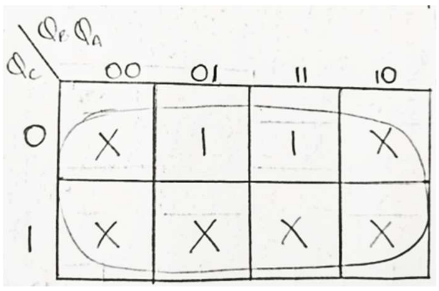
$K_B = Q_A$

J_A:



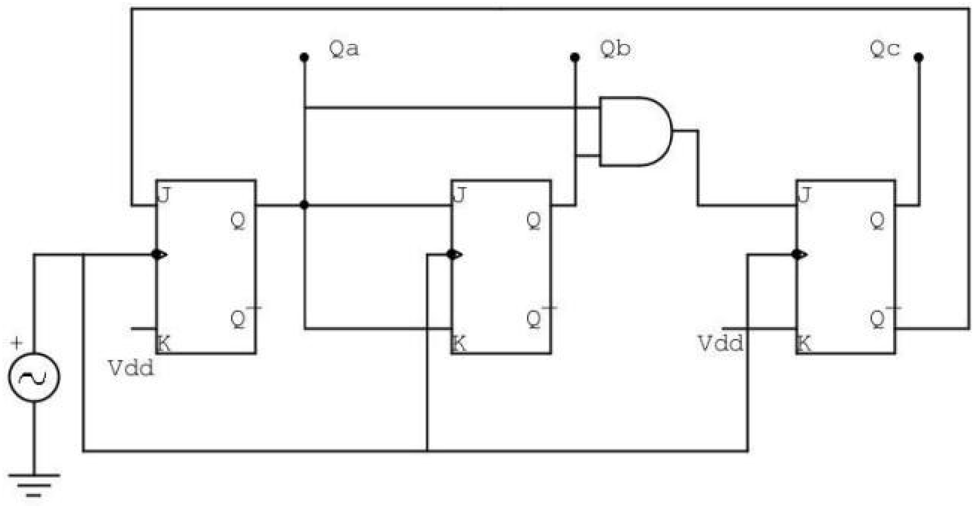
$J_A = Q_c$

K_A:

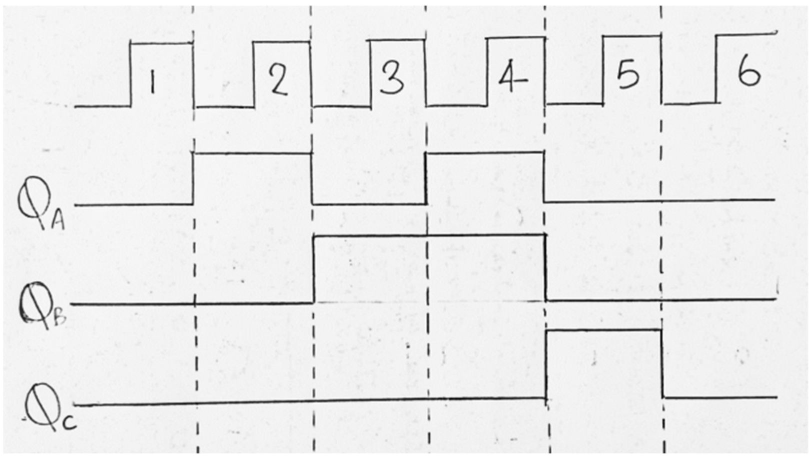


$K_A = 1$

CIRCUIT DIAGRAM



TIMING DIAGRAM:



PROCEDURE:

1. Wire the circuit as per the diagram on the bread board.
2. Provide clock to the circuit from the Function Generator.
3. Verify the count sequence as given in the truth table.

INFERENCE:

A synchronous mod-5 counter was realized using IC 7473, a negative edge triggered JK flip-flop. Its truth value was verified and the timing diagram was plotted.

EXPERIMENT NO. 7b.

SEQUENCE GENERATOR

AIM OF THE EXPERIMENT:

To realize a synchronous counter which generates the sequence 0,2,5,7,0 using IC 7473.

LEARNING OBJECTIVE:

Upon completion of this experiment, students will be able to:

- Understand the construction of mod-N counters.
- Design the counter to produce custom output.

COMPONENTS REQUIRED:

Sl. No	Components	Specifications	Quantity
1.	IC 7473	Dual J-K Flip-Flop with asynchronous clear	2
2.	LED		3
3.	Resistor	330Ω	3

BRIEF DESCRIPTION:

"Mod-N" counter will require "N" number of flip-flops connected together to count a single data bit while providing 2^n different output states, (n is the number of bits). Note that N is always a whole integer value. Then we can see that MOD counters have a modulus value that is an integral power of 2, that is 2, 4, 8, 16 and so on to produce an n-bit counter depending on the number of flip-flops used, and how they are connected, determining the type and modulus of the counter.

DESIGN:

Resistor Design:

Let,

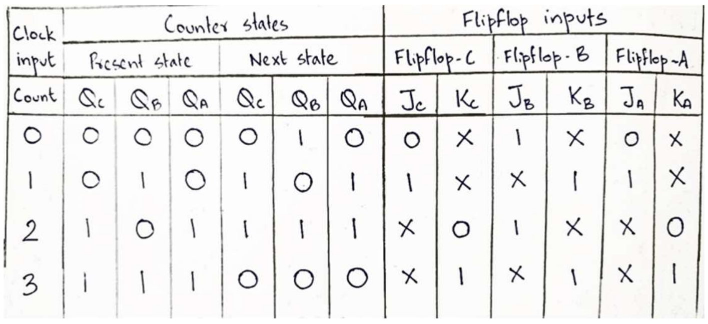
I_D = LED forward current in Amps (found in the LED datasheet)

V_D = LED forward voltage drop in Volts (found in the LED datasheet)

V_{CC} = supply voltage

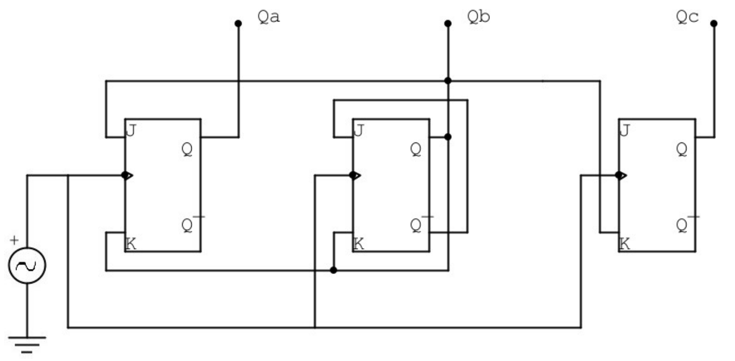
R=(VCC-VD)/ID = (5-1.2)/10=380 Ω ≈ 330Ω

TRUTH TABLE:

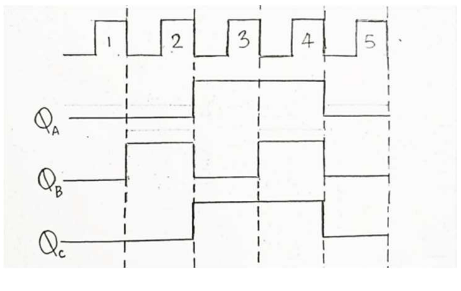


Clock input	Counter states						Flipflop inputs					
	Present state			Next state			Flipflop-C		Flipflop-B		Flipflop-A	
Count	Q_C	Q_B	Q_A	Q_C	Q_B	Q_A	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	1	0	0	X	1	X	0	X
1	0	1	0	1	0	1	1	X	X	1	1	X
2	1	0	1	1	1	1	X	0	1	X	X	0
3	1	1	1	0	0	0	X	1	X	1	X	1

CIRCUIT DIAGRAM



TIMING DIAGRAM:



PROCEDURE:

1. Wire the circuit as per the diagram on the bread board.
2. Provide clock to the circuit from the Function Generator.
3. Verify the count sequence as given in the truth table.

INFERENCE:

A synchronous counter which counts in the sequence 0,2,5,7 is implemented using IC 7473, a negative edge triggered JK flip-flop. Its truth value was verified and the timing diagram was plotted.

EXPERIMENT NO. 8a

STUDY OF SIPO SHIFT REGISTERS

AIM OF THE EXPERIMENT:

To study the working of 4-bit Serial In, Parallel Out shift register implemented using IC 7474.

LEARNING OBJECTIVE:

Upon completion of this experiment, students will be able to:

- Construct SIPO register using IC 7474
- Understand the working of a Serial In, Parallel out Shift Register.

COMPONENTS REQUIRED:

Sl. No	Components	Specifications	Quantity
1.	IC 7474	Dual J-K Flip-Flop	2
2.	LED		4
3.	Resistor	330Ω	4

BRIEF DESCRIPTION:

The **Shift Register** is another type of sequential logic circuit that can be used for the storage orthe 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.

DESIGN:

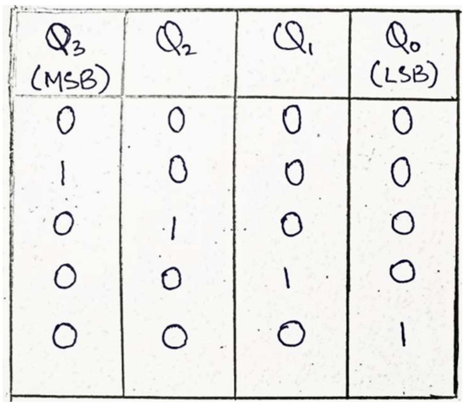
<u>Resistor Design:</u>

Let,

I_D	= LED forward current in Amps (found in the LED datasheet)
V_D	= LED forward voltage drop in Volts (found in the LED datasheet)
V_{CC}	= supply voltage

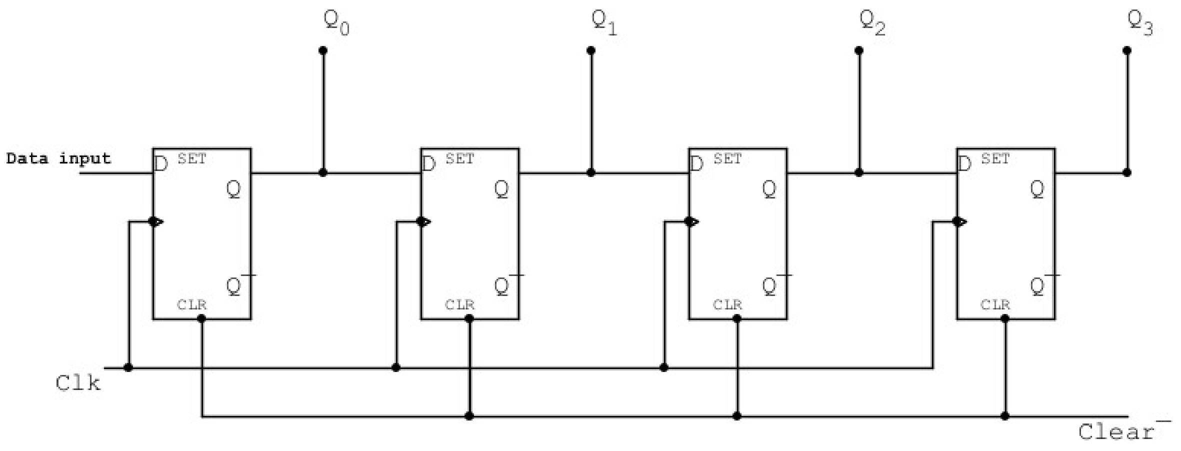
R=(VCC-VD)/ID = (5-1.2)/10=380 Ω ≈ 330Ω

TRUTH TABLE



Q_3 (MSB)	Q_2	Q_1	Q_0 (LSB)
0	0	0	0
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

CIRCUIT DIAGRAM



PROCEDURE:

1. Wire the circuit as per the diagram on the bread board.
2. Preset FF1 and reset all the remaining flip flops.
3. Provide clock signal synchronously.
4. Record the output readings and verify the truth table.

INFERENCE:

A 4-bit Serial In, Parallel Out shift register was realized using JK flip-flop IC 7473 and its functionality was verified.

EXPERIMENT NO. 8b

RING & JOHNSON COUNTER

AIM OF THE EXPERIMENT:

To design a 4-bit Ring counter and Johnson counter using IC 7474 & IC 7495.

LEARNING OBJECTIVE:

Upon completion of this experiment, students will be able to:

- Learn about Ring & Johnson Counter and its applications
- Construct freq/N and freq/2N counters using minimum hardware.

COMPONENTS REQUIRED:

Sl. No	Components	Specifications	Quantity
1.	IC 7474	Dual J-K Flip-Flop	2
2.	IC 7495	AND gate	1
3.	LED		3
4.	Resistor	330Ω	3

BRIEF DESCRIPTION:

The serial movement of data through the register occurs after a preset number of clock cycles thereby allowing the SISO register to act as a sort of time delay circuit to the original input data signal. On connecting the output of this shift register back to its input so that the output from the last flip-flop, QD becomes the input of the first flip-flop, we would then have a closed loop circuit that “recirculates” the same bit of DATA around a continuous loop for every state of its sequence, and this is the principal operation of a Ring counter. Thereby looping the output back to the input, (feedback) we can convert a standard shift register circuit into a ring counter.

4-bit Ring Counter the synchronous Ring Counter is preset so that exactly one data bit in the registeris set to logic “1” with all other bits reset to “0”. To achieve this, a “CLEAR” signal is firstly applied to all the flip-flops together in order to “RESET” their outputs to a logic “0” level and then a “PRESET” pulse is applied to the input of the first flip-flop (FFA) before the clock pulsesare applied. This then places a single logic “1” value into the circuit of the ring counter. So on each successive clock pulse, the counter circulates the same data bit between the four flip-flops over and over again around the “ring” every fourth clock cycle. But in order to cycle the data correctly around the counter we must first “load” the counter with a suitable data pattern as all logic “0’s” or all logic “1’s” outputted at each clock cycle would make the ring counter invalid. This type of data movement is called “rotation”, and like the previous shift register, the effect of the movement of the data bit from left to right through a ring counter can be presented graphically as follows along with its timing diagram. Since the ring counter example shown above has four distinct states, it is also known as a “modulo-4” or “mod-4” counter with each flip-flop output having a frequency value equal to one-fourth or a quarter (1/4) that of the main clock frequency. The “MODULO” or “MODULUS” of a counter is the number of states the counter counts or sequences through before repeating itself and a ring counter can be made to output any modulo number. A “mod-n” ring counter will require “n” number of flip-flops connected together to circulate a single data bit providing “n” different output states. **Johnson Counter** also known as Twisted Ring Counter is another basic application of shift registers with a feedback. Here the feedback is given from the inverted output of the last flip flop to the input of the first flip-flop. It consists of four flip-flops FF0, FF1, FF2 and FF3. Here the inverted output of the last flip-flop FF3 is given as feedback to the input of the first flip-flop FF0. Here, at first four logic zeros will be passed to the flip-flops. When clock pulses are given “1000”, “1100”, “1110”, “1111”, “0111”, “0011”, “0001”, “0000” outputs will be obtained and the sequence will repeat for the next clock pulses.

DESIGN:

Resistor Design:

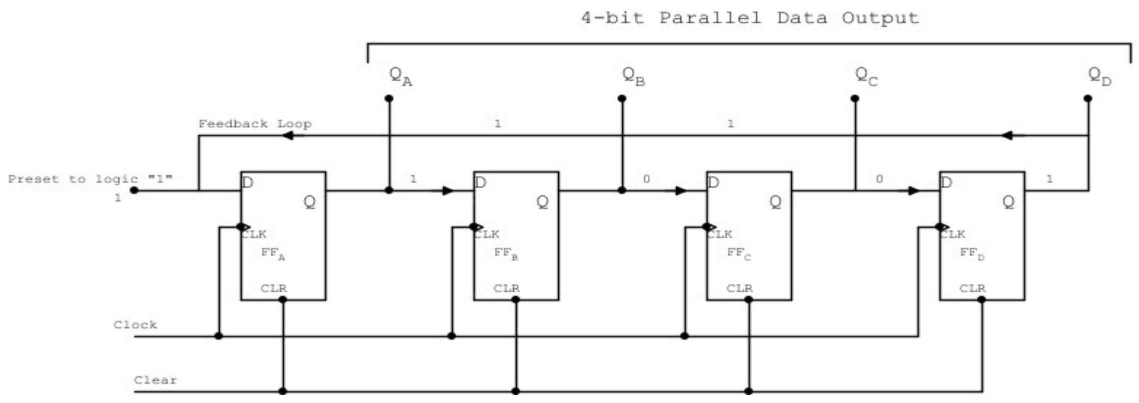
Let,

I_D	= LED forward current in Amps (found in the LED datasheet)
V_D	= LED forward voltage drop in Volts (found in the LED datasheet)
V_{CC}	= supply voltage

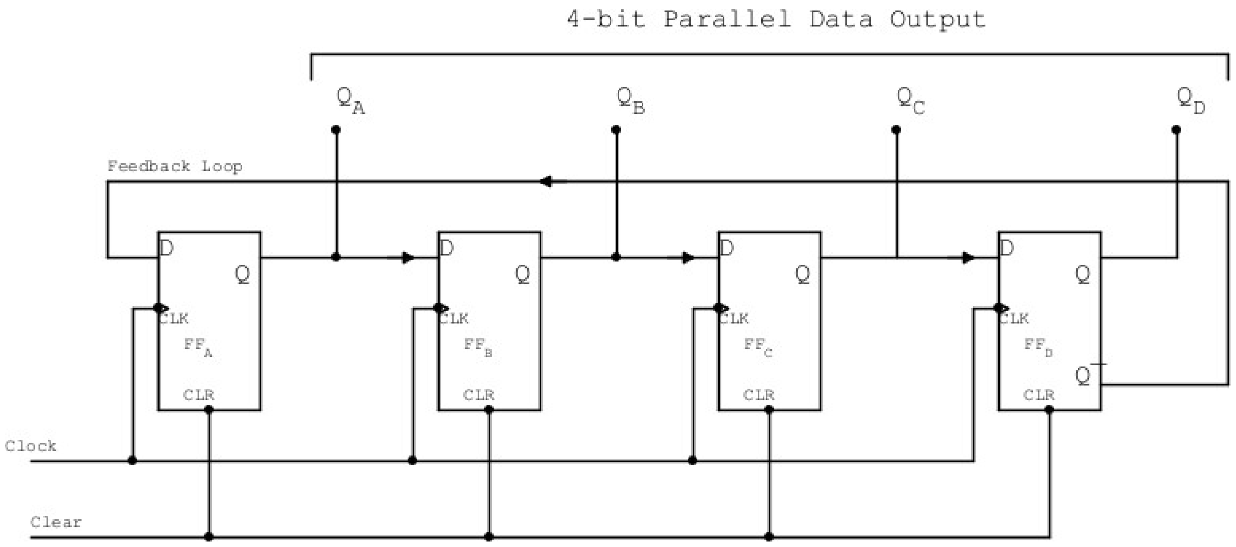
R=(VCC-VD)/ID = (5-1.2)/10=380 Ω ≈ 330Ω

CIRCUIT DIAGRAM

a) Ring Counter using IC 7474



b) Johnson Counter using IC 7474



PROCEDURE:

1. Make connections as per circuit diagram
2. Apply 1 Hz clock input synchronously
3. Pull all the clear pins to HIGH
4. For Ring operation, Preset FFA for one clock pulse and then pull it HIGH
5. Record the output readings and verify the truth table.

INFERENCE:

A 4-bit ring and Johnson counter was realized using JK flip-flop IC 7473 and its functionality was verified.

EXPERIMENT NO. 9

MULTIPLEXERS AND DEMULTIPLEXERS

AIM OF THE EXPERIMENT:

To realize a 4:1 MUX and 1:4 DEMUX using basic gates.

LEARNING OBJECTIVE:

Upon completion of this experiment, students will be able to:

- Implement Multiplexers and Demultiplexers

COMPONENTS REQUIRED:

Sl. No	Components	Specifications	Quantity
1.	IC 7408	Quad 2 input AND gate	1
2.	IC 7432	Quad 2 input OR gate	1
3.	IC 7404	Quad 2 input NOT gate	1
4.	LED		1
5.	Resistor	330Ω	1

BRIEF DESCRIPTION:

Multiplexer (MUX) is a device that selects one of several analog or digital input signals and forwards the selected input into a single line. A multiplexer of 2 m inputs has m select lines,which are used to select which input line to send to the output. Multiplexers are mainly used increase the amount of data that can be sent over the network within a certain amount of time and bandwidth. A multiplexer is also called a data selector.

Conversely, a demultiplexer (DEMUX) is a device taking a single input signal and selecting one of many data-output-lines, which is connected to the single input. A multiplexer is often used with a complementary demultiplexer on the receiving end. The conversion from one code to another is common in digital systems.

DESIGN:

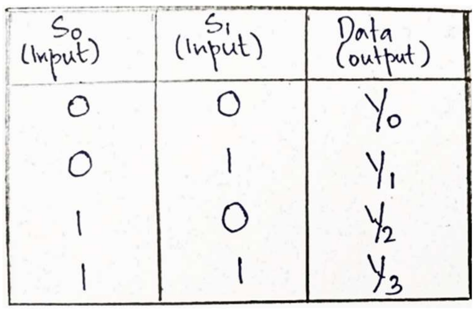
Resistor Design:

Let, I_D = LED forward current in Amps (found in the LED datasheet)

V_D = LED forward voltage drop in Volts (found in the LED datasheet)
V_{CC} = supply voltage
R=(VCC-VD)/ID = (5-1.2)/10=380 **Ω** ≈ 330Ω

4:1 MUX using logic gates

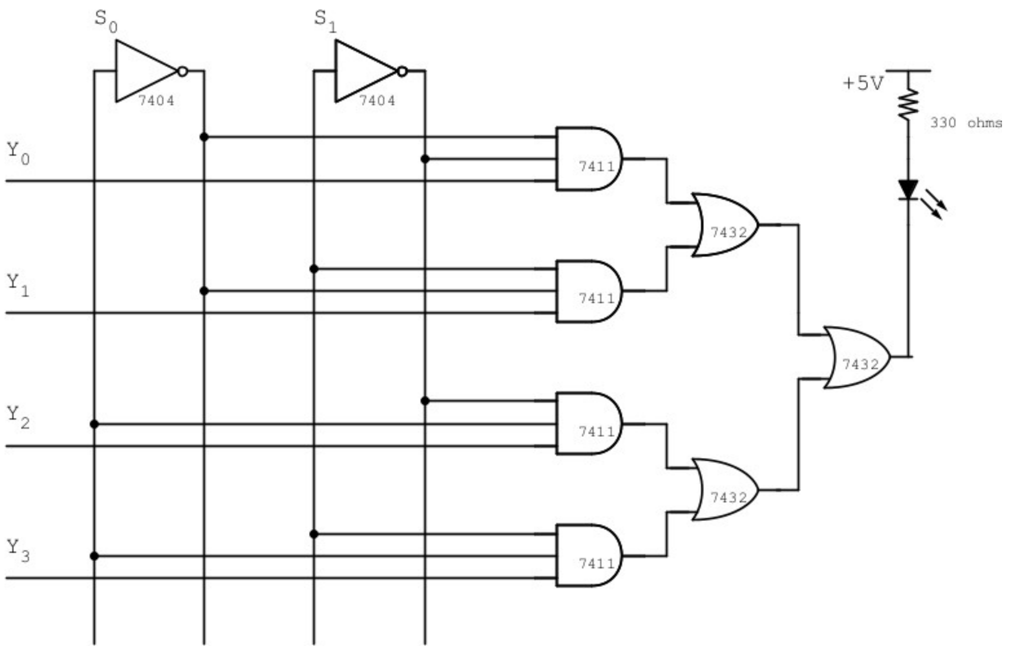
TRUTH TABLE:



S_0 (Input)	S_1 (Input)	Data (output)
0	0	Y_0
0	1	Y_1
1	0	Y_2
1	1	Y_3

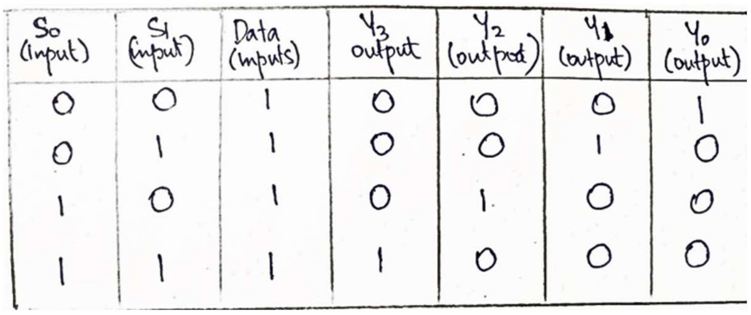
Data= $S'_0S'_1Y_0+S'_0S1Y_1+S_0S'_1Y_2+S_0S_1Y_3$

CIRCUIT DIAGRAM



1:4 DEMUX using logic gates

TRUTH TABLE:



S0 (input)	S1 (input)	Data (inputs)	Y3 output	Y2 (output)	Y1 (output)	Y0 (output)
0	0	1	0	0	0	1
0	1	1	0	0	1	0
1	0	1	0	1	0	0
1	1	1	1	0	0	0

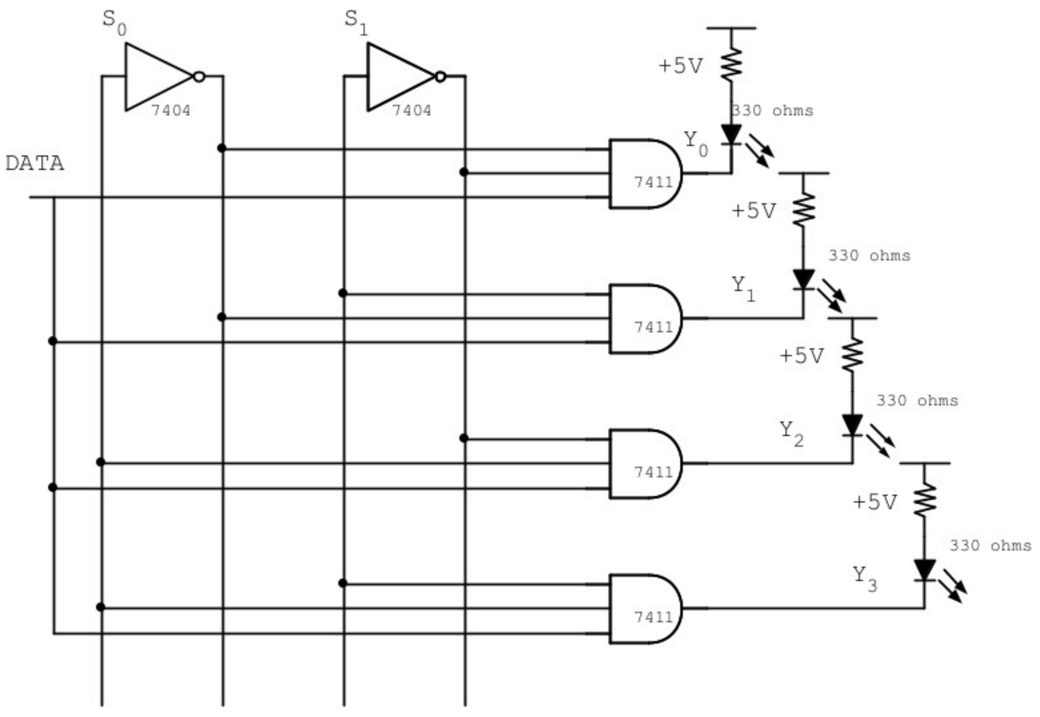
$Y_0 = S_1'S_0'$ (DATA)

$Y_1 = S_1S_0'$ (DATA)

$Y_2 = S_1'S_0$ (DATA)

$Y_3 = S_1S_0$ (DATA)

CIRCUIT DIAGRAM



PROCEDURE:

1. Wire the circuit as per the diagram on the bread board.
2. Apply various input combinations and observe output for each one.
3. Verify the truth table for each input/output combination.

INFERENCE:

A 4:1 MUX and 1:4 DEMUX realized using basic gates and its functionality was verified.

EXPERIMENT NO. 10

MAGNITUDE COMPARATOR

AIM OF THE EXPERIMENT:

To design and implement a circuit that accepts two 2-bit numbers and generates three output values based on the magnitude of the numbers.

LEARNING OBJECTIVE:

Upon completion of this experiment, students will be able to:

- Implement 2 bit magnitude comparator using logic gates.

COMPONENTS REQUIRED:

Sl. No	Components	Specifications	Quantity
1.	IC 7486	Quad 2-input XOR gate	1
2.	IC 7408	Quad 2-input AND gate	1
3.	IC 7432	Quad 2-input OR gate	1
4.	IC 7404	Hex Inverter	1
5.	LED		3
6.	Resistor	330Ω	3

BRIEF DESCRIPTION:

In digital system, comparison of two numbers is an arithmetic operation that determines if one number is greater than, equal to, or less than the other number. So comparator is used for this purpose. Magnitude comparator is a combinational circuit that compares two numbers, A and B, and determines their relative magnitudes. The outcome of comparison is specified by three binary variables that indicate whether
A>B, A=B, or A<B.

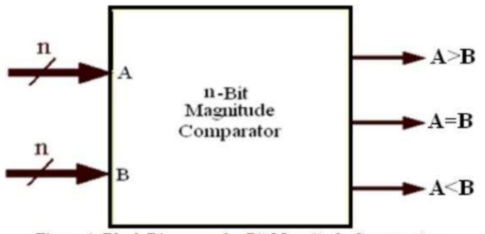
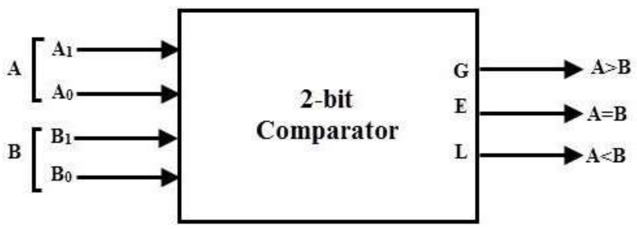


Figure 1. Block Diagram of n-Bit Magnitude Comparator

2 Bit Magnitude Comparator:
For a 2-bit comparator we have four inputs A1A0 and B1B0 and three outputs:
E (is 1 if two numbers are equal)
G (is 1 when A > B) and
L (is 1 when A < B)



The logic for a 2-bit magnitude comparator. Let the two 2-bit numbers be $A=A_1A_0$ and $B=B_1B_0$.

1. If $A_1=1$ and $B_1=0$, then A>B or
2. If A_1 and B_1 coincide and $A_0=1$ and $B_0=0$, then A>B. So the logic expression for A > B is

$$A > B: G = A_1B_1 \quad (A_1 \odot B_1)A_0B_0$$

1. If $A_1=0$ and $B_1=1$, then A<B or

2. If A_1 and B_1 coincide and $A_0=0$ and $B_0=1$, then A<B. So the logic expression for A > B is
$A < B: L = A_1\bar{B}_1 + (A_1 \odot B_1)\ A_0\bar{B}_0$

3. If A_1 and B_1 coincide and $A_0=0$ and $B_0=0$, then A=B. So the logic expression for A = B is

$$A = B: E = (A_1 \odot B_1)\ (A_0 \odot B_0)$$

DESIGN:

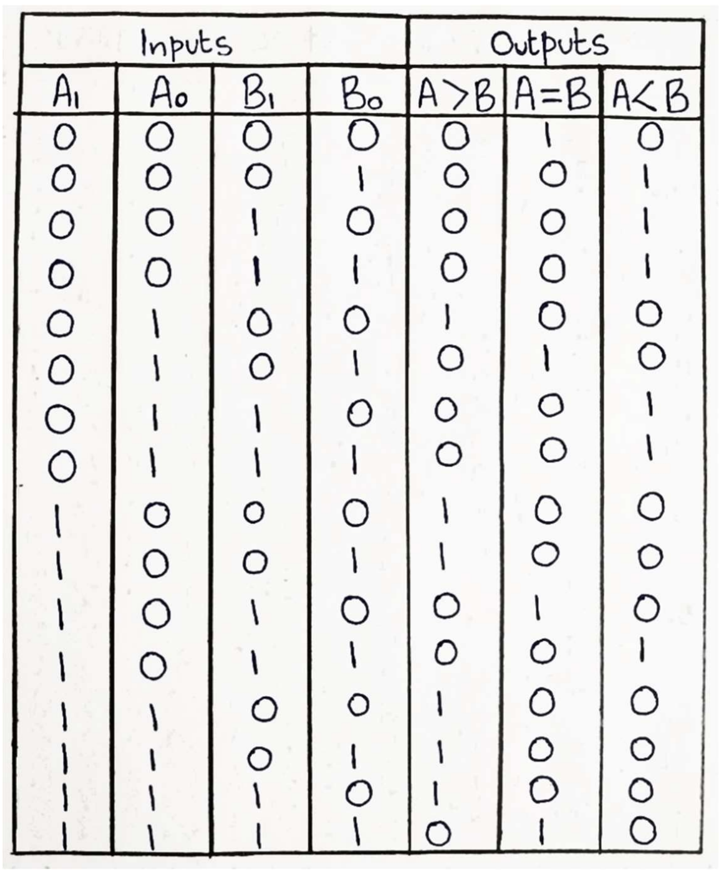
Resistor Design:

Let,

I_D	= LED forward current in Amps (found in the LED datasheet)
V_D	= LED forward voltage drop in Volts (found in the LED datasheet)
V_{CC}	= supply voltage

$R=(V_{CC}-V_D)/I_{D}=(5-1.2)/10=380\ \Omega \approx 330\Omega$

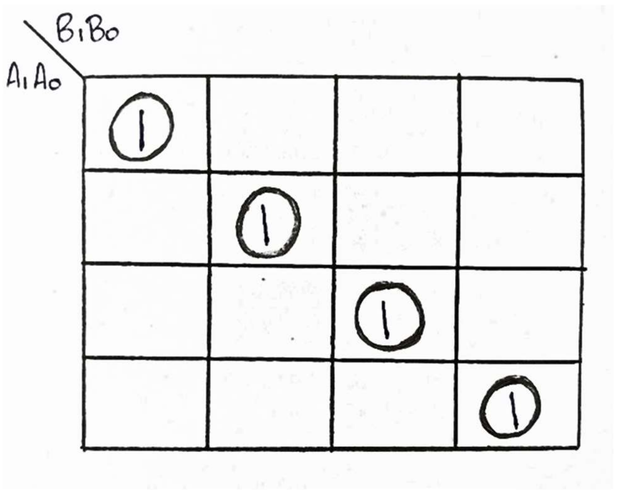
TRUTH TABLE:



Inputs				Outputs		
A_1	A_0	B_1	B_0	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

K map simplification

For A=B



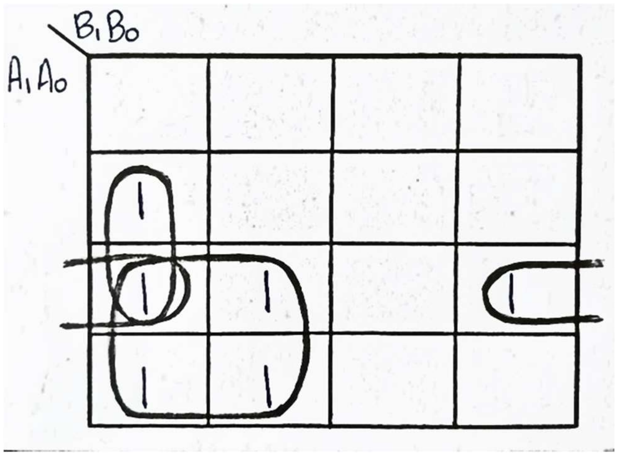
E(A=B) = $A_1'A_0'B_1'B_0' + A_1'A_0B_1'B_0 + A_1A_0B_1B_0 + A_1A_0'B_1B_0'$

= $A_1'B_1'(A_0'B_0' + A_0B_0) + A_1B_1(A_0B_0 + A_0'B_0')$

= $(A_0B_0 + A_0'B_0')(A_1B_1 + A_1'B_1')$

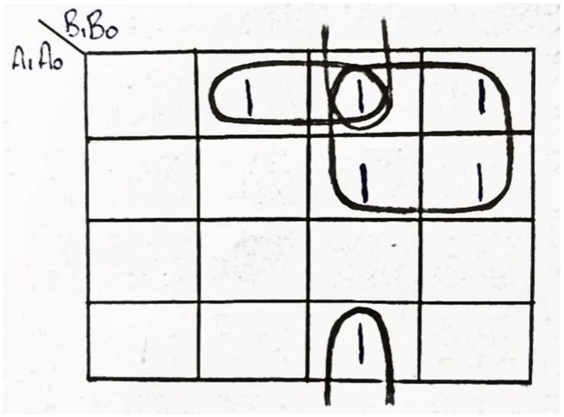
= $(A_0 \odot B_0)(A_1 \odot B_1)$

For A>B



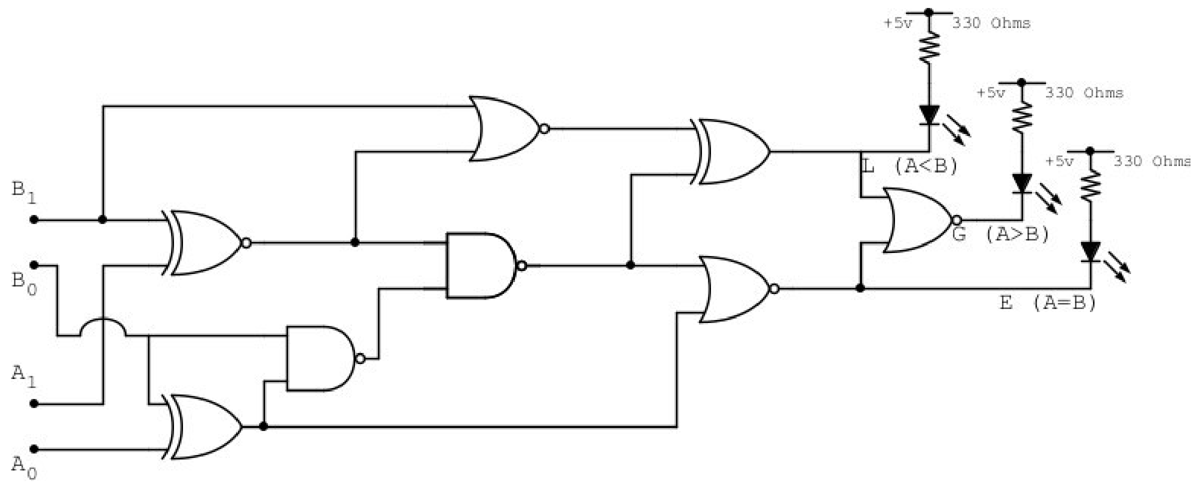
G(A>B) = $A_1B_1' + A_0B_1'B_0' + A_1A_0B_0'$

For A<B



L(A<B) = $A_1'B_1 + A_0'B_1B_0 + A_1'A_0'B_0$

CIRCUIT DIAGRAM



PROCEDURE:

1. Wire the circuit as per the diagram on the bread board.

2. Observe output for each combinations.

3. Verify the truth table for each input/ output combination.

INFERENCE:

A two bit magnitude comparator was implemented using logic gates and its truth value was verified.