Lab Manual

for

Digital Computer Principles

Diploma In Computer Engineering

3rd Semester

by

SITTTR

Kalamassery

STATE INSTITUTE OF TECHNICAL TEACHERS TRAINING AND RESEARCH

GENERAL INSTRUCTIONS

Rough record and Fair record are needed to record the experiments conducted in the laboratory. Rough records are needed to be certified immediately on completion of the experiment. Fair records are due at the beginning of the next lab period. Fair records must be submitted as neat, legible, and complete.

INSTRUCTIONS TO STUDENTS FOR WRITING THE FAIR RECORD

In the fair record, the index page should be filled properly by writing the corresponding experiment number, experiment name , date on which it was done and the page number.

On the **right side** page of the record following has to be written:

1. **Title**: The title of the experiment should be written in the page in capital letters.

2. In the left top margin, experiment number and date should be written.

3. **Aim**: The purpose of the experiment should be written clearly.

4.**Apparatus/Tools/Equipments/Components used:** A list of the Apparatus/Tools /Equipments /Components used for doing the experiment should be entered.

5. **Principle**: Simple working of the circuit/experimental set up/algorithm should be written.

6. **Procedure:** steps for doing the experiment and recording the readings should be briefly described(flow chart/programs in the case of computer/processor related experiments)

7. **Results**: The results of the experiment must be summarized in writing and should be fulfilling the aim.

8. **Inference** : Inference from the results is to be mentioned.

On the **Left side** page of the record following has to be recorded:

1. **Circuit/Program**: Neatly drawn circuit diagrams/experimental set up.

2. **Design**: The design of the circuit/experimental set up for selecting the components

should be clearly shown if necessary.

3. **Observations:** i) Data should be clearly recorded using Tabular Columns.

ii) Unit of the observed data should be clearly mentioned

iii) Relevant calculations should be shown. If repetitive calculations are needed, only show a sample calculation and summarize the others in a table.

4. **Graphs :** Graphs can used to present data in a form that show the results obtained, as one or more of the parameters are varied. A graph has the advantage of presenting large

amounts of data in a concise visual form. Graph should be in a square format.

GENERAL RULES FOR PERSONAL SAFETY

1. Always wear tight shirt/lab coat , pants and shoes inside workshops.
2. REMOVE ALL METAL JEWELLERY since rings, wrist watches or bands, necklaces, etc. make excellent electrodes in the event of accidental contact with electric power sources.
3. DO NOT MAKE CIRCUIT CHANGES without turning off the power.
4. Make sure that equipment working on electrical power are grounded properly.
5. Avoid standing on metal surfaces or wet concrete. Keep your shoes dry.
6. Never handle electrical equipment with wet skin.
7. Hot soldering irons should be rested in its holder. Never leave a hot iron unattended.

8. Avoid use of loose clothing and hair near machines and avoid running around inside lab .

TO PROTECT EQUIPMENT AND MINIMIZE MAINTENANCE:

DO: 1. SET MULTIRANGE METERS to highest range before connecting to an unknown source.

2. INFORM YOUR INSTRUCTOR about faulty equipment so that it can be sent for repair.

DO NOT: 1. Do not MOVE EQUIPMENT around the room except under the supervision of an instructor.

Index

Cycle I - Implementation of basic logic gates

DIGITAL COMPUTER PRINCIPLES LAB

Lab Manual

The Breadboard

The breadboard consists of two terminal strips and two bus strips (often broken in the centre). Each bus strip has two rows of contacts. Each of the two rows of contacts is a node. That is, each contact along a row on a bus strip is connected together (inside the breadboard). Bus strips are used primarily for power supply connections, but are also used for any node requiring a large number of connections. Each terminal strip has 60 rows and 5 columns of contacts on each side of the centre gap. Each row of 5 contacts is a node.

You will build your circuits on the terminal strips by inserting the leads of circuit components into the contact receptacles and making connections with 22-26 gauge wire. There are wire cutter/strippers and a spool of wire in the lab. It is a good practice to wire +5V and 0V power supply connections to separate bus strips.

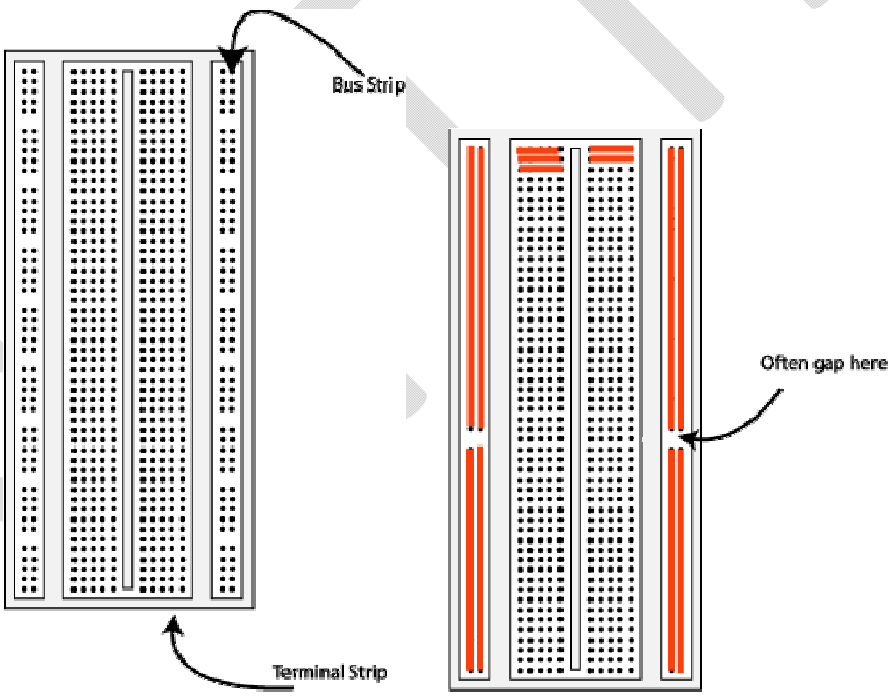


Fig . The breadboard. The lines indicate connected holes.

The 5V supply **MUST NOT BE EXCEEDED** since this will damage the ICs (Integrated circuits) used during the experiments. Incorrect connection of power to the ICs could result in them exploding or becoming very hot - with the **possible serious injury occurring to the people working on the experiment! Ensure that**

the power supply polarity and all components and connections are correct <u>before</u> switching on power.

Building the Circuit

Throughout these experiments we will use TTL chips to build circuits. The steps for wiring a circuit should be completed in the order described below:

1. Turn the power (Trainer Kit) off before you build anything!
2. Make sure the power is off before you build anything!
3. Connect the +5V and ground (GND) leads of the power supply to the power and ground bus strips on your breadboard.
4. Plug the chips you will be using into the breadboard. Point all the chips in the same direction with pin 1 at the upper-left corner. (Pin 1 is often identified by a dot or a notch next to it on the chip package)
5. Connect +5V and GND pins of each chip to the power and ground bus strips on the breadboard.
6. Select a connection on your schematic and place a piece of hook-up wire between corresponding pins of the chips on your breadboard. It is better to make the short connections before the longer ones. Mark each connection on your schematic as you go, so as not to try to make the same connection again at a later stage.
7. Get one of your group members to check the connections, **before you turn the power on**.
8. If an error is made and is not spotted before you turn the power on. Turn the power off immediately before you begin to rewire the circuit.
9. At the end of the laboratory session, collect you hook-up wires, chips and all equipment and return them to the demonstrator.
10. Tidy the area that you were working in and leave it in the same condition as it was before you started.

Cycle I

Implementation of basic logic gates

Exp.No…1……. Name of the Experiment: Gates using basic components

Aim: Implement logic gates using basic components

Objectives: To study the construction of logic gates (**OR, AND, NOT, NOR, NAND** gates etc.) using discrete components and verify their truth tables

Components: Bread board/ Kit , Resistors 10k,1k,220ohms, Transistors 2N2222(NPN), Diodes 1N 4001, Connecting wires

Theory:

a) An OR gate is a logic circuit with two or more inputs and one output. The output of an OR gate is LOW only when all of its inputs are LOW. For all other possible input combinations, the output is HIGH

b) The output of an AND gate is HIGH only when all of its inputs are in the HIGH state. In all other cases, the output is LOW

c) A NOT gate is a one-input, one-output logic circuit whose output is always the complement of the input. That is, a LOW input produces a HIGH output, and vice versa.

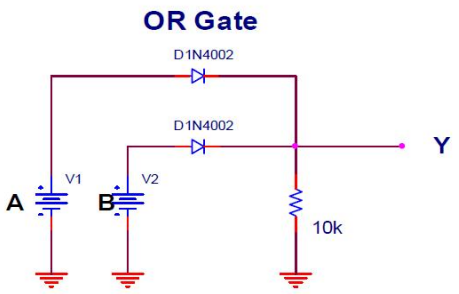
d) The output of a NOR gate is a logic '1' when all its inputs are logic '0'. For all other input combinations, the output is a logic '0'.

e) The output of a NAND gate is a logic '0' when all its inputs are a logic '1'. For all other input combinations, the output is a logic '1'.

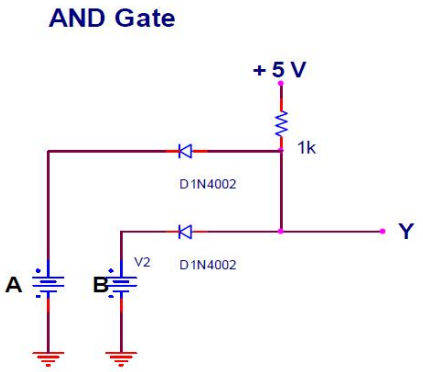
Circuit diagrams

Circuit Diagrams:

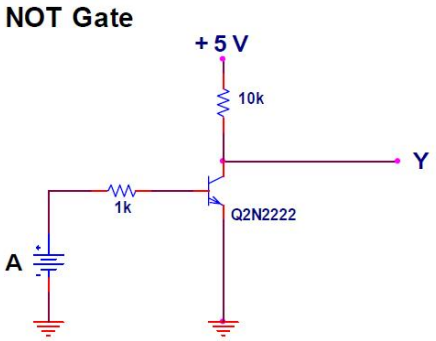
TRUTH TABLE



A	B	Y
0v	0v	0v
0v	5v	5v
5v	0v	5v
5v	5v	5v

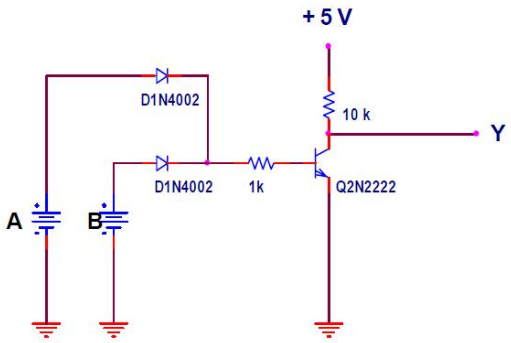


A	B	Y
0v	0v	0v
0v	5v	0v
5v	0v	0v
5v	5v	5v



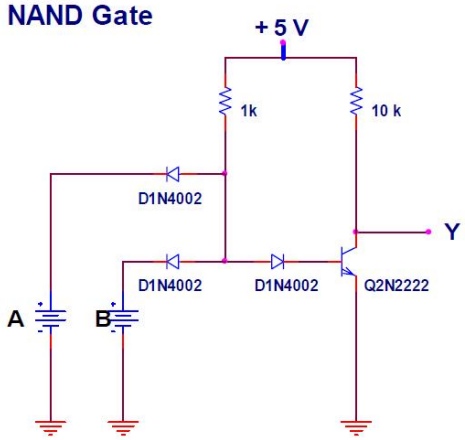
A	Y
0v	5v
5v	0v

NOR Gate



A	B	Y
0v	0v	5v
0v	5v	0v
5v	0v	0v
5v	5v	0v

NAND Gate



A	B	Y
0v	0v	5v
0v	5v	5v
5v	0v	5v
5v	5v	0v

Procedure

1. Connections are made as per the circuit diagram
2. Switch on the power supply
3. Apply different combinations of inputs and observe the outputs; compare the outputs with the truth tables

Result: Different logic gates are constructed and their truth tables are verified.

Exp.No…2……. Name of the Experiment: Gates using IC

Aim: Verify the Logic behavior of various IC gates: 7408, 7432, 7404, 7400, 7402, 7486

Objectives: To study the working of logic gates (**OR, AND, NOT, NOR, NAND, XOR** gates etc.) using IC and verify their truth tables

Components: Bread board/ Kit , 7408, 7432, 7404, 7400, 7402, 7486

Theory:

a) An OR gate is a logic circuit with two or more inputs and one output. The output of an OR gate is LOW only when all of its inputs are LOW. For all other possible input combinations, the output is HIGH

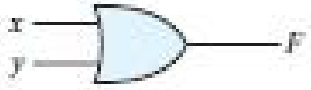
b) The output of an AND gate is HIGH only when all of its inputs are in the HIGH state. In all other cases, the output is LOW

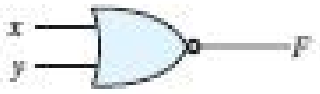
c) A NOT gate is a one-input, one-output logic circuit whose output is always the complement of the input. That is, a LOW input produces a HIGH output, and vice versa.

d) The output of a NOR gate is a logic '1' when all its inputs are logic '0'. For all other input combinations, the output is a logic '0'.

e) The output of a NAND gate is a logic '0' when all its inputs are a logic '1'. For all other input combinations, the output is a logic '1'.

Circuit and Truth table

Name	Graphic symbol	Algebraic function	Truth table
AND		$F = x \cdot y$	x y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = x + y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
Inverter		$F = x'$	x | F 0 | 1 1 | 0

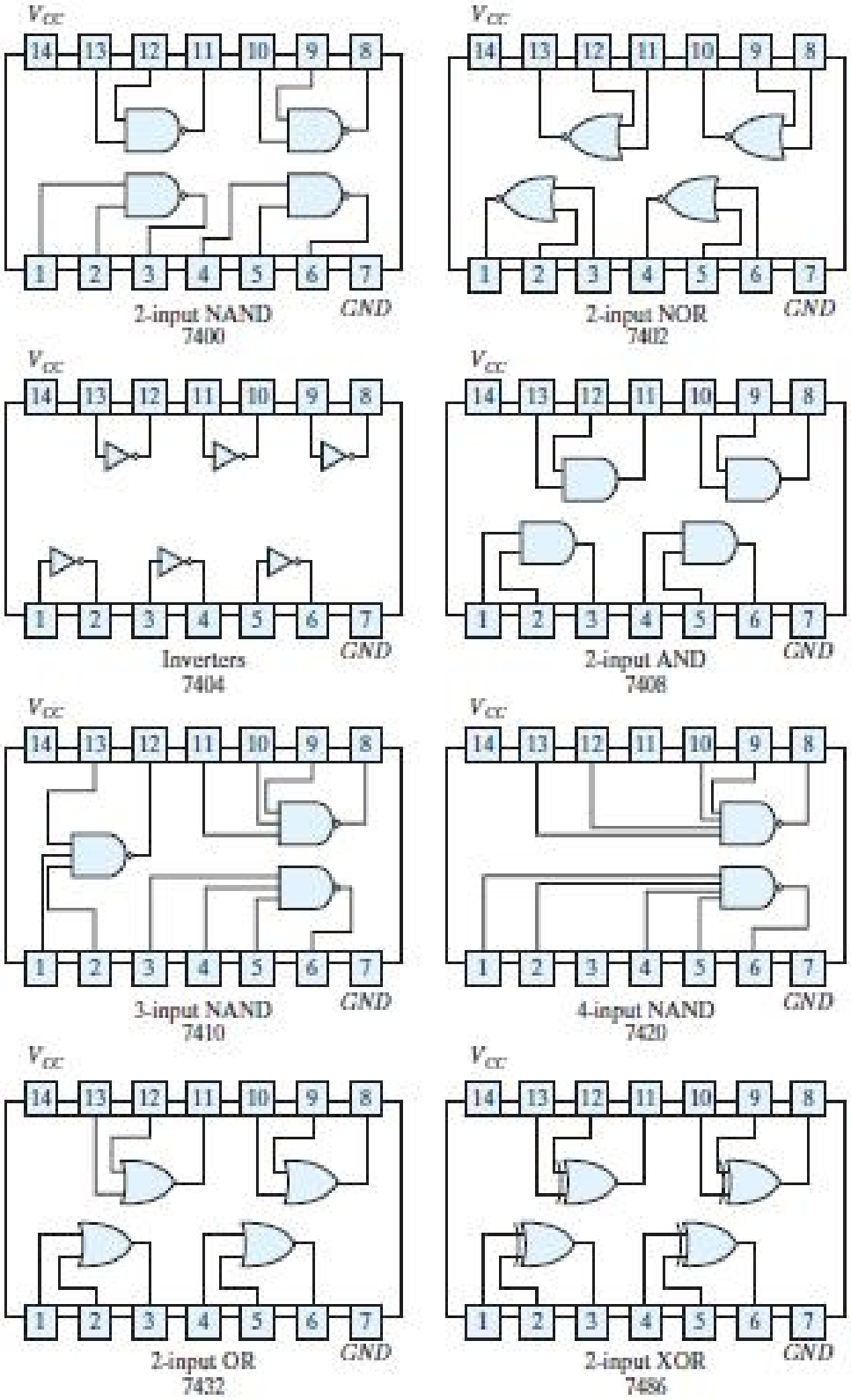
NAND		$F = (xy)'$	x y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = (x + y)'$	x y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0

Procedure

a. Test all the ICs manually/ using IC tester.
b. Connect VCC and the ground.
c. Connect the appropriate pins to the input and output LEDs and switches.
d. Verify the truth table with respect to the clock.

Result : Different logic gates and their truth tables are verified.

Useful IC Pin details



Exp.No…3……. ***Name of the Experiment:*** Demonstration of universal gates

Aim: Implement basic gates using Universal Gates (NAND & NOR)

Objectives: To study the design and implementation of logic gates (**OR, AND, NOT, NOR, NAND, XOR** gates etc.) using Universal Gates (NAND & NOR) and verify their truth tables

Components: Bread board/ Kit , 7400, 7402

Theory:

a) An OR gate is a logic circuit with two or more inputs and one output. The output of an OR gate is LOW only when all of its inputs are LOW. For all other possible input combinations, the output is HIGH

b) The output of an AND gate is HIGH only when all of its inputs are in the HIGH state. In all other cases, the output is LOW

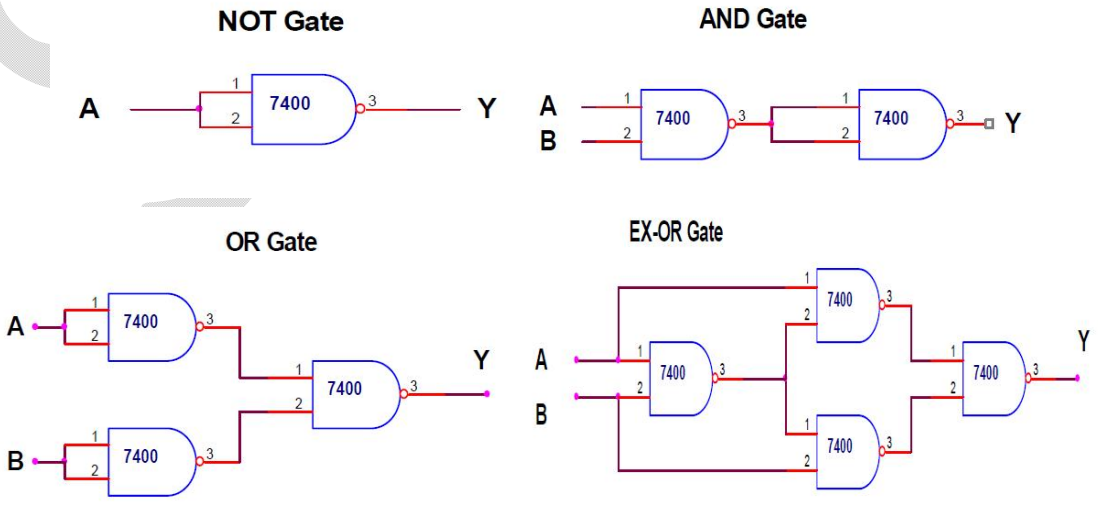
c) A NOT gate is a one-input, one-output logic circuit whose output is always the complement of the input. That is, a LOW input produces a HIGH output, and vice versa.

d) The output of a NOR gate is a logic '1' when all its inputs are logic '0'. For all other input combinations, the output is a logic '0'.

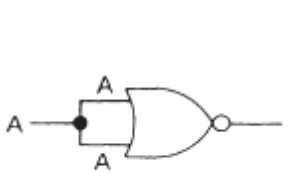
e) The output of a NAND gate is a logic '0' when all its inputs are a logic '1'. For all other input combinations, the output is a logic '1'.

REALIZATION OF GATES USING NAND

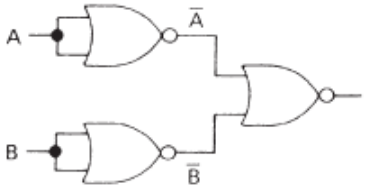
Circuit diagram



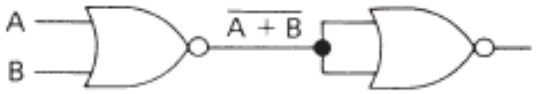
REALIZATION OF GATES USING NOR



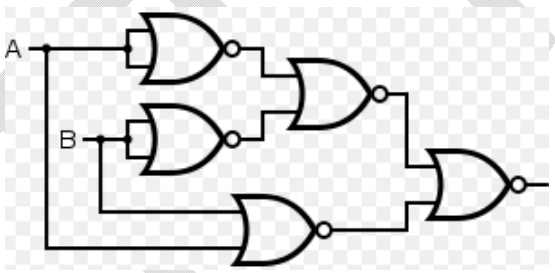
NOT gate circuit



AND Gate



OR Gate



XOR Gate

Procedure

a. Test all the ICs manually/ using IC tester.
b. Connect VCC and the ground.
c. Connect the appropriate pins to the input and output LEDs and switches.
d. Verify the truth table with respect to the clock.

Result Different logic gates are constructed and their truth tables are verified.

Exp.No…4……. ***Name of the Experiment:*** SOP & POS

Aim: Plot the following Boolean function in a Map as well as implement in a logic diagram

F= A'D + BD +B'C + AB'D

Objectives: To Study the Simplification of Boolean Functions SOP & POS forms (Demonstrates the relationship between a Boolean Function and the corresponding logic diagram – using Map reduction method)

Components: Bread board/ Kit , 7408,7404,7432

Theory:

a) Convert the given function in Standard SOP Form., Enter into K-Map and reduce it.

b) Convert the given function in Standard POS Form., Enter into K-Map and reduce it.

c) Verify that (a) and (b) is equal

Convert the given function in to Standard SOP Form

F= A'D + BD +B'C + AB'D

Since it is in SOP convert to Std SOP

*) Add the variable B for the missing terms

A'D=A'D(B+B')=A'DB+A'DB'

*) Add the variable A for the missing terms

BD=BD(A'+A)=A'BD+ABD

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

*) Add the variable C for the missing terms

A'DB=A'DB(C'+C)= A'DBC'+ A'DBC

A'DB'=ADB'(C'+C)= ADB'C' +ADB'C

A'BD=A'BD(C'+C)= A'BDC'+ A'BDC

ABD=ABD(C'+C)= ABDC'+ ABDC

AB'D=AB'D(C'+C)= AB'DC'+ AB'DC

*) Add the variable D for the missing terms

A'B'C=A'B'C(D'+D)= A'B'CD'+ A'B'CD

AB'C=AB'C(D'+D)= AB'CD'+AB'CD

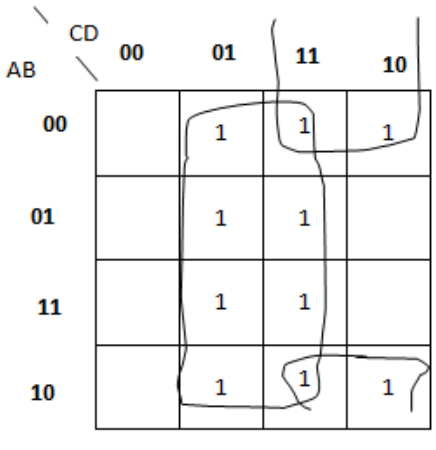
*) combine the above four variable terms all together

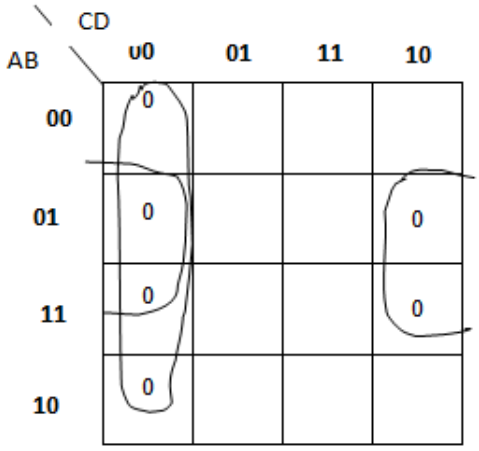
F=A'BC'D+A'BCD+A'B'C'D+A'B'CD+A'BC'D+A'BCD+ABC'D +ABCD+AB'C'D+AB'CD+A'B'CD'+ A'B'CD +AB'CD' +AB'CD

Ie F= m(1,2,3,5,7,9,10,11,13,15)

Ie f= M(0,4,6,8,12,14)

Enter into K-map and minimise





Sop F min=D+B'C **POS F max=(C+D)(B'+D)**

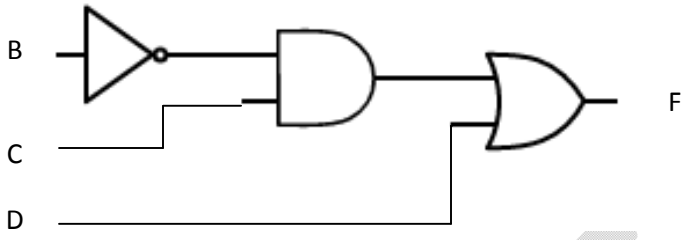
Truth table for Sop F min=D+B'C

C	B	D	B'C	D+B'C
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	0	0
1	1	1	0	1

POS F max=(C+D)(B'+D)

C	B	D	C+D	B'+D	(C+D)(B'+D)
0	0	0	0	1	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	1	0	0
1	1	1	1	1	1

The above Truth table shows both are equal but in SOP minimized equation needs 3 gate and POS minimized equation needs 4 gates; so here SOP minimized equation is more economical



Procedure

1. Connections are made as per the circuit diagram
2. Switch on the power supply
3. Apply different combinations of inputs and observe the outputs; compare the outputs with the truth tables

Result : Different logic circuit are constructed and their truth tables are verified.

Cycle 2

Combinational Circuits

Exp.No…5……. ***Name of the Experiment:*** Parity bit generator

Aim: Design, construct, and test circuit that generates parity bit from four message bits

Objectives: To study the construction and design of the Combinational Circuits

Components: Bread board/ Kit , 7486

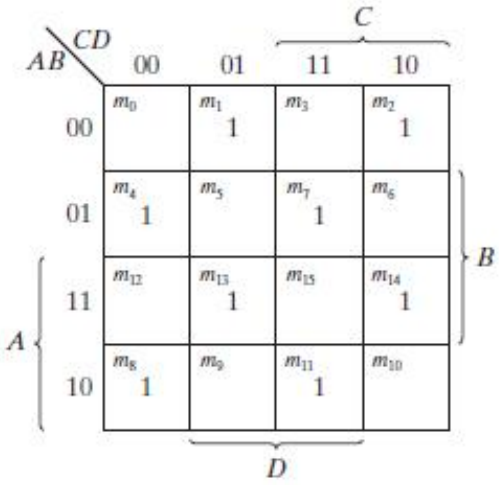
Theory: A parity bit is an extra bit included with a binary message to make the number of 1's either odd or even. The message, including the parity bit, is transmitted and then checked at the receiving end for errors. An error is detected if the checked parity does not correspond with the one transmitted. The circuit that generates the parity bit in the transmitter is called a *parity generator*

Circuit diagram and truth table

Truth table

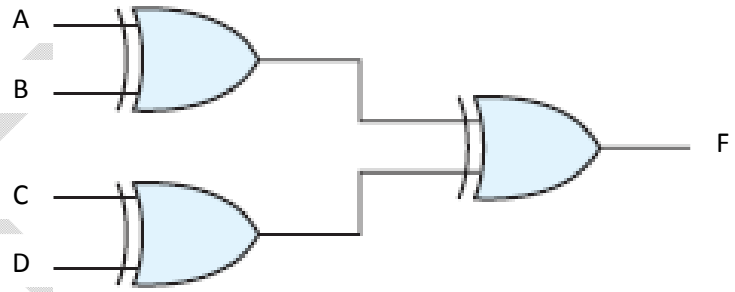
A	*B*	*C*	*D*	*P*
0	*0*	*0*	*0*	*0*
0	*0*	*0*	*1*	*1*
0	*0*	*1*	*0*	*1*
0	*0*	*1*	*1*	*0*
0	*1*	*0*	*0*	*1*
0	*1*	*0*	*1*	*0*
0	*1*	*1*	*0*	*0*
0	*1*	*1*	*1*	*1*
1	*0*	*0*	*0*	*1*
1	*0*	*0*	*1*	*0*
1	*0*	*1*	*0*	*0*
1	*0*	*1*	*1*	*1*
1	*1*	*0*	*0*	*0*
1	*1*	*0*	*1*	*1*
1	*1*	*1*	*0*	*1*
1	*1*	*1*	*1*	*0*

K-map



$$F = A \oplus B \oplus C \oplus D$$

Circuit diagram



Procedure

1. Connections are made as per the circuit diagram
2. Switch on the power supply
3. Apply different combinations of inputs and observe the outputs; compare the outputs with the truth tables

Result : Different logic circuit are constructed and their truth tables are verified.

Exp.No…6……. ***Name of the Experiment:*** Code converter

Aim: Design a circuit that converts a 1) Gray code to binary, 2) Decoder for a binary digit to BCD, and 3) a seven segment indicator

Objectives: To study the design, implementation and the working of code converters.

Components: Bread board/ Kit , 7486,7408, 7432, 7404, 7730, 7447

Theory:

Gray code to binary,

a) Most significant bit (B_1) is same as the most significant bit in Gray Code ($B_1 = G_1$)
b) To find next bit perform Ex-OR (Exclusive OR) between the Current binary bit and previous bit.
   $G_n = B_n$ (Ex-OR) B_{n-1}

Binary digit to BCD

A binary code that distinguishes among 10 elements must contain at least four bits, but 6 out of the 16 possible combinations remain unassigned. Different binary codes can be obtained by arranging four bits into 10 distinct combinations..

A decimal number in BCD is the same as its equivalent binary number only when the number is between 0 and 9. A BCD number greater than 10 looks different from its equivalent binary number, even though both contain 1's and 0's. Moreover, the binary combinations 1010 through 1111 are not used and have no meaning in BCD

Seven segment indicator

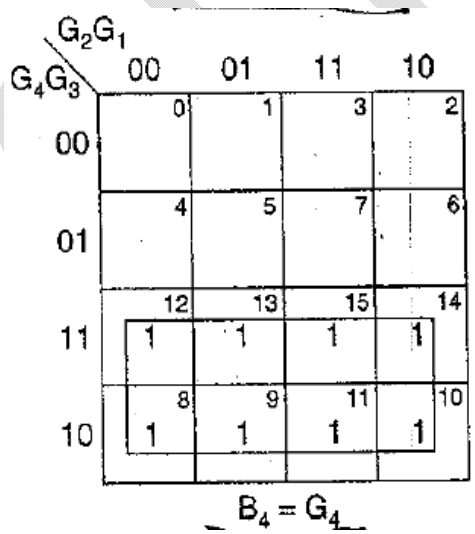
A seven-segment indicator is used to display any one of the decimal digits 0 through 9. Usually, the decimal digit is available in BCD. A BCD-to-seven-segment decoder accepts a decimal digit in BCD and generates the corresponding seven-segment code

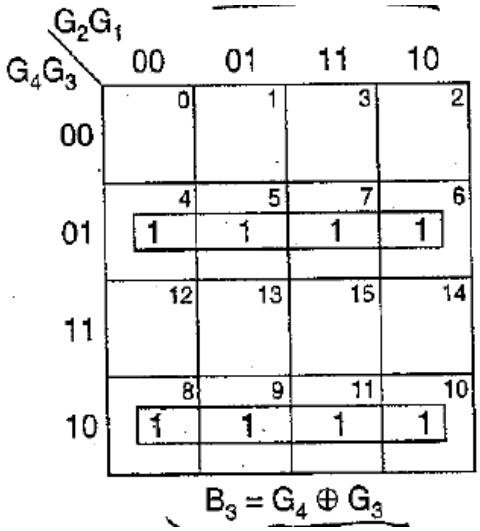
Circuit and truth table

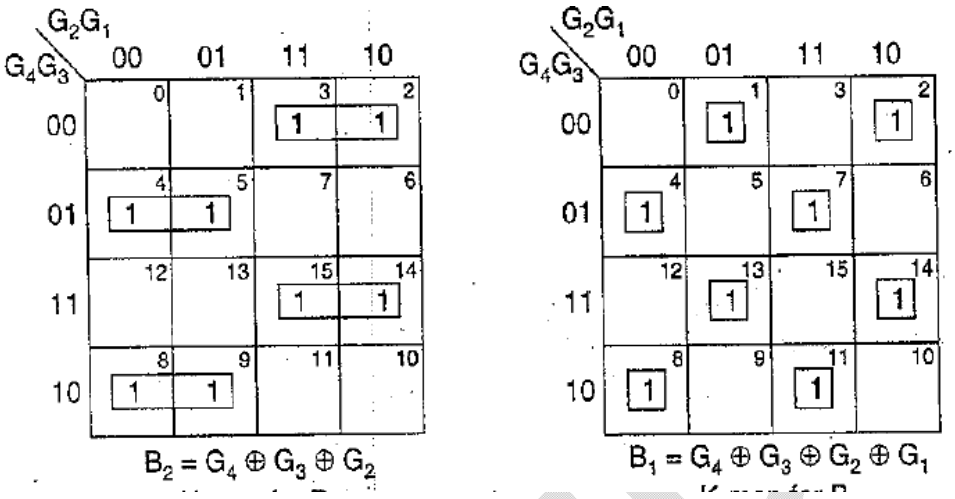
Gray code to binary

Truth table

4-bit Gray				4-bit binary			
G_4	G_3	G_2	G_1	B_4	B_3	B_2	B_1
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

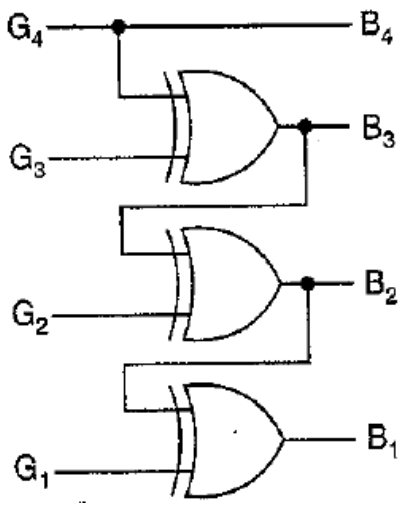






K-map

Circuit diagram



4 bit Binary to BCD Converter
Truth Table

Decimal	4-bit binary				BCD output				
	B_4	B_3	B_2	B_1	A	B	C	D	E
0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	1
2	0	0	1	0	0	0	0	1	0
3	0	0	1	1	0	0	0	1	1
4	0	1	0	0	0	0	1	0	0
5	0	1	0	1	0	0	1	0	1
6	0	1	1	0	0	0	1	1	0
7	0	1	1	1	0	0	1	1	1
8	1	0	0	0	0	1	0	0	0
9	1	0	0	1	0	1	0	0	1
10	1	0	1	0	1	0	0	0	0
11	1	0	1	1	1	0	0	0	1
12	1	1	0	0	1	0	0	1	0
13	1	1	0	1	1	0	0	1	1
14	1	1	1	0	1	0	1	0	0
15	1	1	1	1	1	0	1	0	1

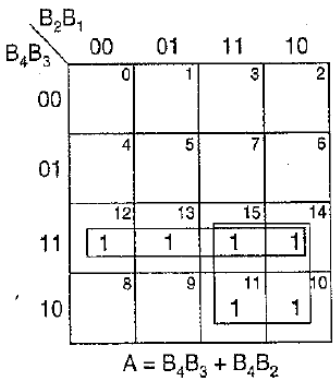
$A = \Sigma\, m(10, 11, 12, 13, 14, 15)$

$B = \Sigma\, m(8, 9)$

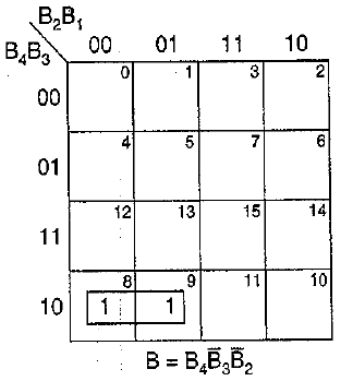
$C = \Sigma\, m(4, 5, 6, 7, 14, 15)$

$D = \Sigma\, m(2, 3, 6, 7, 12, 13)$

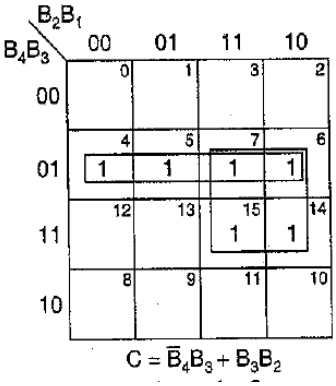
$E = \Sigma\, m(1, 3, 5, 7, 9, 11, 13, 15)$



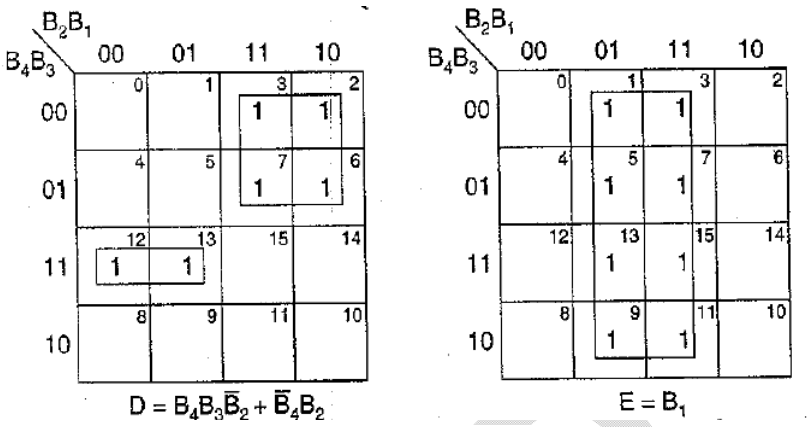
$A = B_4B_3 + B_4B_2$



$B = B_4\bar{B}_3\bar{B}_2$

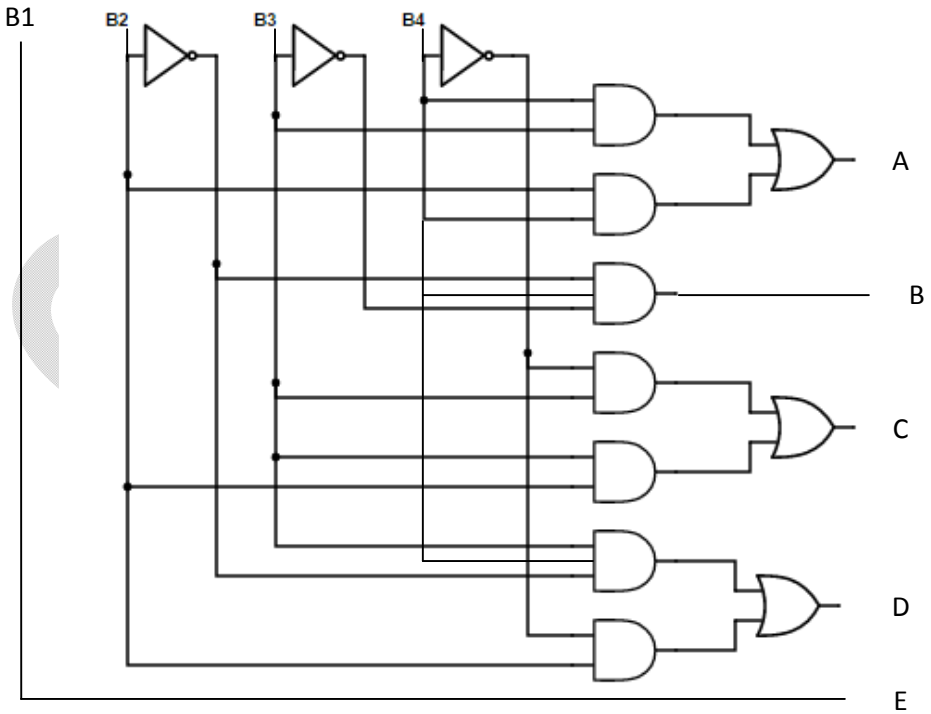


$C = \bar{B}_4B_3 + B_3B_2$



$$A = B_4B_3 + B_4B_2$$
$$B = B_4\bar{B}_3\bar{B}_2$$
$$C = \bar{B}_4B_3 + B_3B_2$$
$$D = B_4B_3\bar{B}_2 + \bar{B}_4B_2$$
$$E = B_1$$

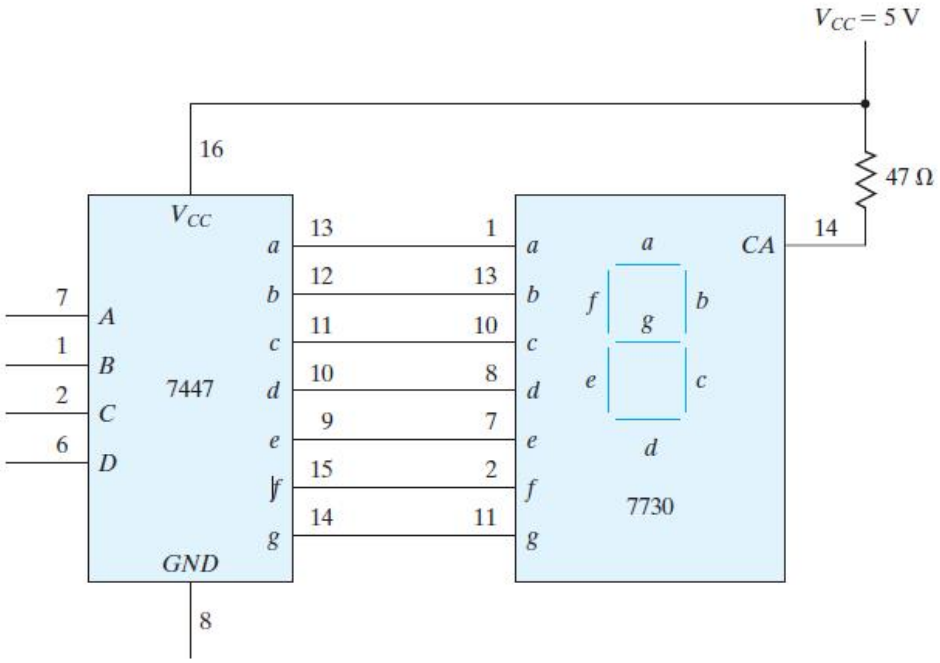
Circuit



Seven segment indicator

Construct the circuit shown in Fig. . Apply the four-bit BCD digits through four switches, and observe the decimal display from 0 to 9. Inputs 1010 through

1111 have no meaning in BCD. Depending on the decoder, these values may cause either a blank or a meaningless pattern to be displayed. Observe and record the output patterns of the six unused input combinations.



Procedure

1. Connections are made as per the circuit diagram
2. Switch on the power supply
3. Apply different combinations of inputs and observe the outputs; compare the outputs with the truth tables

Result : Different logic circuit are constructed and their truth tables are verified.

Exp.No…7……. Name of the Experiment: Half-adder & Full-adder
Aim: Design, construct, and test a half-adder & a full-adder
Objectives: To study the Design, construction, and the working of adders using desecrate Gates
Components: Bread board/ Kit , 7486, 7408, 7400 & 7432
Theory:

Half adder: circuit needs two binary inputs and two binary outputs. The input variables designate the augend and addend bits; the output variables produce the sum and carry. We assign symbols *x* and *y* to the two inputs and *S* (for sum) and *C* (for carry) to the outputs. The truth table for the half adder is listed in Table

Full adder : A full adder is a combinational circuit that forms the arithmetic sum of three bits. It consists of three inputs and two outputs. Two of the input variables, denoted by *x* and *y* , represent the two significant bits to be added. The third input, *z* , represents the carry from the previous lower significant position.

Circuit and truth table

Half Adder

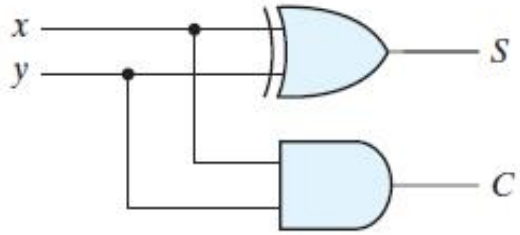
Truth table

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

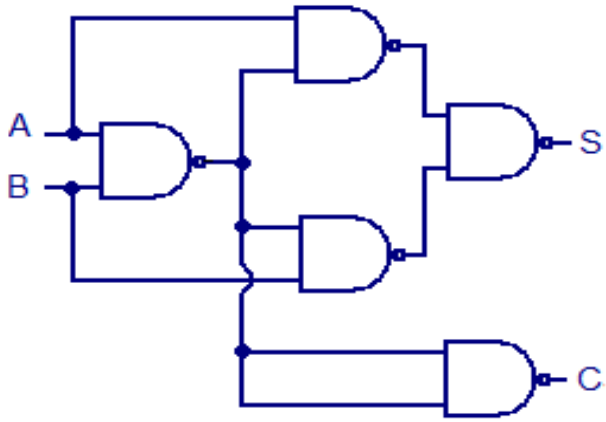
The simplified Boolean functions for the two outputs can be obtained directly from the truth table.

$$S = x'y + xy'$$
$$C = xy$$

Circuit diagram



Using NAND gate



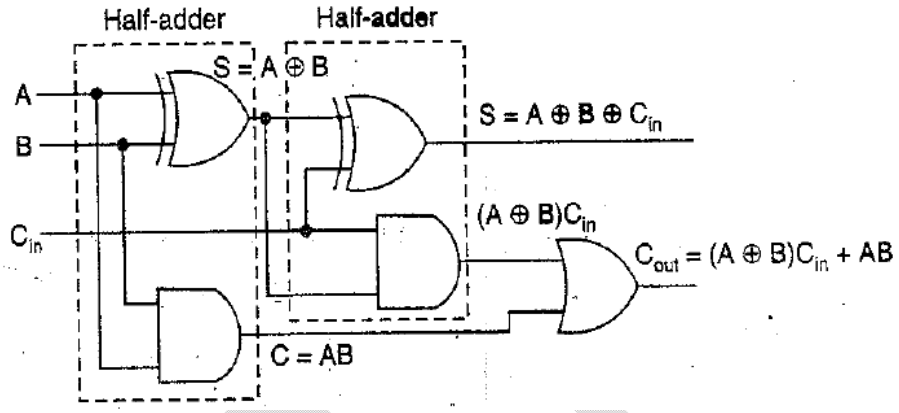
Full adder
Truth Table

x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

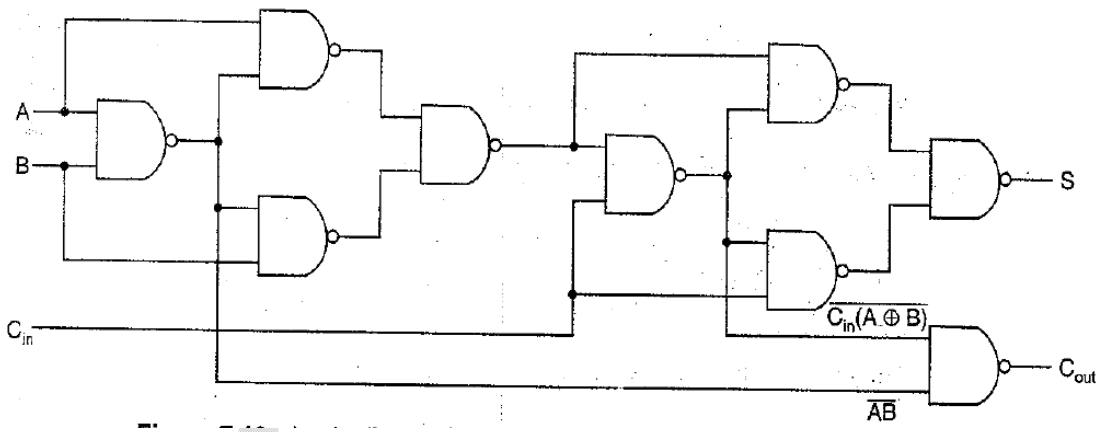
$$S = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in}$$
$$= (A\bar{B} + \bar{A}B)\bar{C}_{in} + (AB + \bar{A}\bar{B})C_{in} = (A \oplus B)\bar{C}_{in} + \overline{(A \oplus B)}C_{in} = A \oplus B \oplus C_{in}$$

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

Circuit



Circuit using NAND gate



Procedure

1. Connections are made as per the circuit diagram
2. Switch on the power supply
3. Apply different combinations of inputs and observe the outputs; compare the outputs with the truth tables

Result : Different logic circuit are constructed and their truth tables are verified.

Exp.No…8……. Name of the Experiment: Binary parallel adder , Adder-Subtractor, and a Magnitude Comparator

Aim: Implement a circuit using a four-bit binary parallel adder (IC 7483) implement Adder- Subtractor, and a Magnitude Comparator

Objectives: To study the implementation of parallel adder, Adder- Subtractor, and a Magnitude Comparator using IC 7483

Components: Bread board/ Kit , 7404, 7408 & 7483

Theory:

Parallel adder : IC type 7483 is a four-bit binary parallel adder. The pin assignment is shown in Fig.. The 2 four-bit input binary numbers are *A1* through *A4* and *B1* through *B4* . The four-bit sum is obtained from *S1* through *S4* . *C0* is the input carry and *C4* the output carry.

Adder–Subtractor: Two binary numbers can be subtracted by taking the 2's complement of the subtrahend and adding it to the minuend. The 2's complement can be obtained by taking the 1's complement and adding 1. To perform *A* - *B*, we complement the four bits of *B*, add them to the four bits of *A,* and add 1 through the input carry.

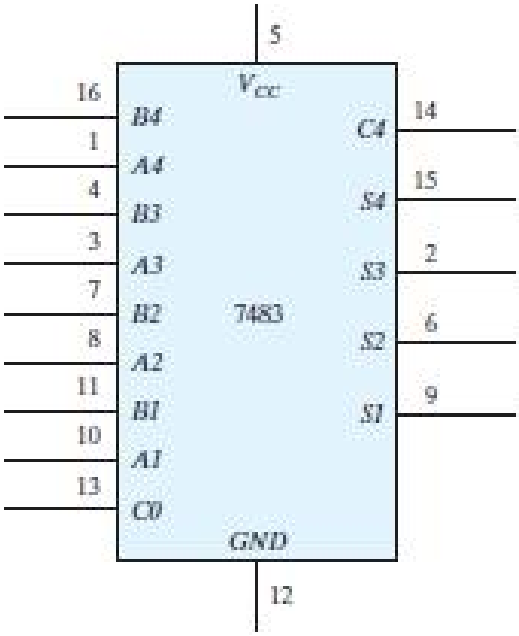
Magnitude Comparator : The comparison of two numbers is an operation that determines whether one number is greater than, equal to, or less than the other number. Two numbers, *A* and *B*, can be compared by first subtracting *A* - *B* . If the output in *S* is equal to zero, then *A* = *B*. The output carry from *C4* determines the relative magnitudes of the numbers: When *C4* = 1, *A* Ú *B*; when *C4* = 0, *A* 6 *B*; and when *C4* = 1 and *S* _ 0, *A* 7 *B*. It is necessary to supplement the subtractor circuit of Fig. to provide the comparison logic. This is done with a combinational circuit that has five inputs— *S1* through *S4* and *C4* —and three outputs, designated by *x, y*, and *z*, so that

x = 1 if *A* = *B* (*S* = 0000)
y = 1 if *A* < *B* (*C4* = 0)
z = 1 if *A* > *B* (*C4* = 1 and *S* _ 0000)

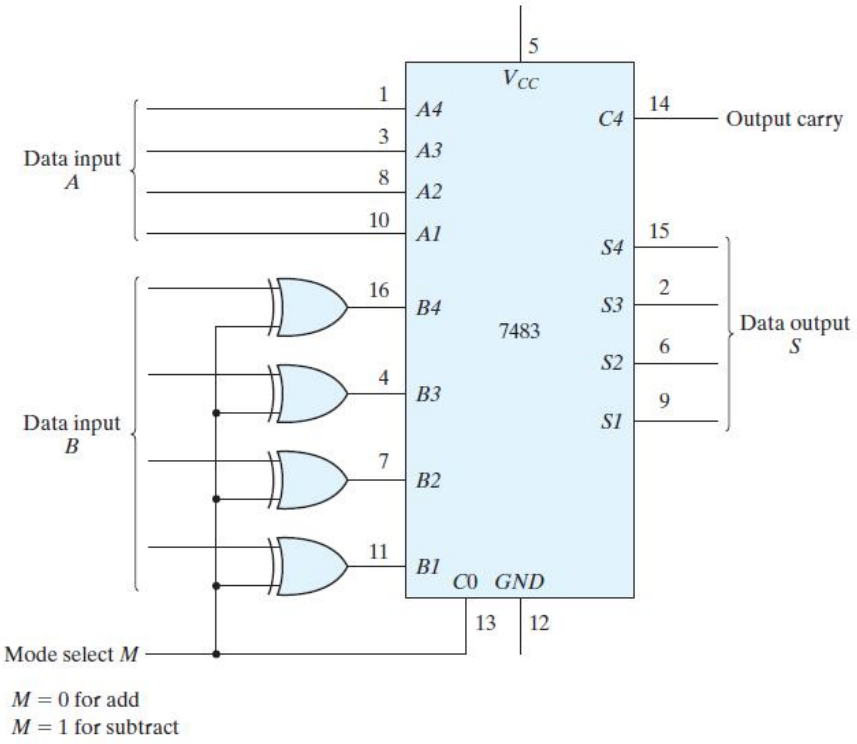
The combinational circuit can be implemented with the 7404 and 7408 ICs. Construct the comparator circuit and test its operation. Use at least two sets of numbers for *A* and *B* to check each of the outputs *x, y*, and *z* .

Circuit

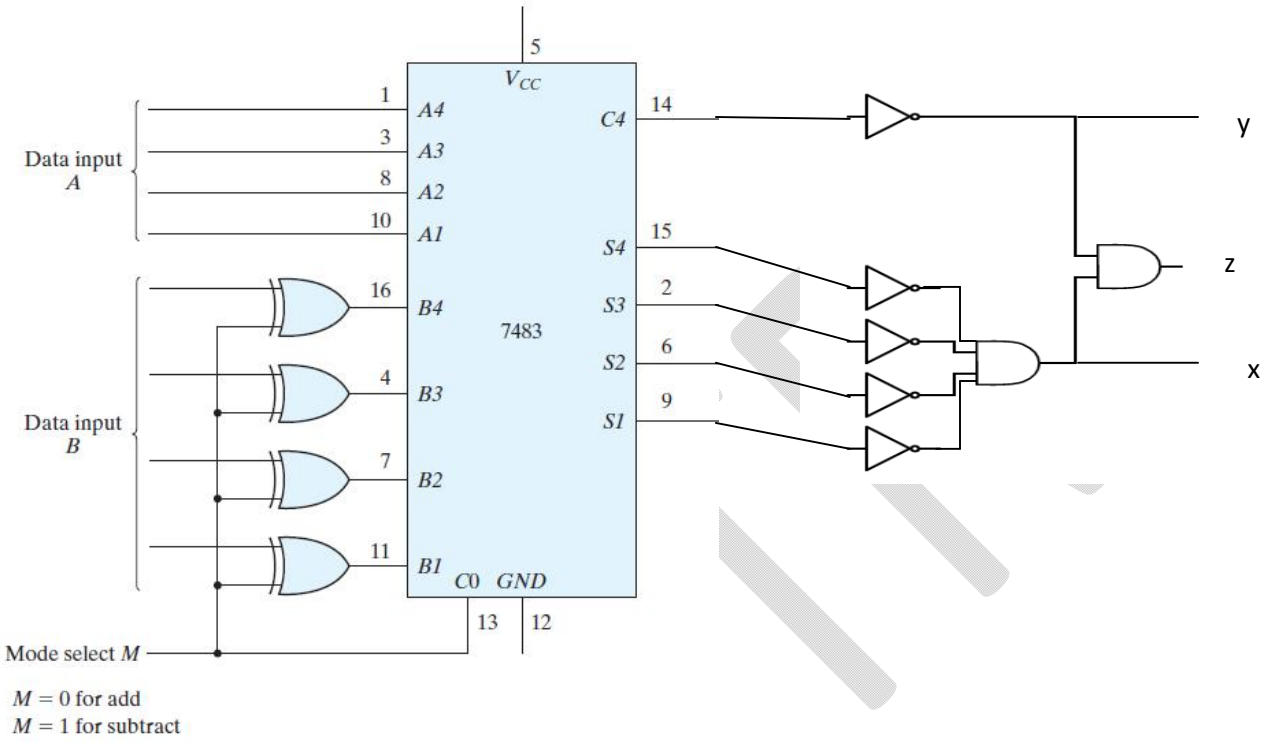
Parallel adder



Adder–Subtractor



Magnitude Comparator



Procedure

1. Connections are made as per the circuit diagram
2. Switch on the power supply
3. Apply different combinations of inputs and observe the outputs; compare the outputs with the truth tables

Result : Different logic circuit are constructed and their truth tables are verified.

Cycle 3-

Synchronous sequential logic

Exp.No…9……. Name of the Experiment: Flip-Flops
Aim: Construct, Test, and investigate the operation of SR Latch, D Latch, Master-Slave Flip-Flop, Edge-Triggered Flip-Flop, IC Flip-Flops(using IC 7476, and 7474)
Objectives: To study the Construction and the working of SR Latch, D Latch, Master-Slave Flip-Flop, Edge-Triggered Flip-Flop, IC Flip-Flops
Components: Bread board/ Kit , 7476, 7474, 7400, 7402 &7404
Theory:

SR Latch,: The *SR* latch is a circuit with two cross-coupled NOR gates or two cross-coupled NAND gates, and two inputs labeled *S* for set and *R* for reset. The *SR* latch constructed with two cross-coupled NOR gates is shown in Fig. . The latch has two useful states. When output $Q = 1$ and $Q\_ = 0$, the latch is said to be in the *set state* . When $Q = 0$ and $Q\_ = 1$, it is in the *reset state* . Outputs Q and $Q\_$ are normally the complement of each other. However, when both inputs are equal to 1 at the same time, a condition in which both outputs are equal to 0 (rather than be mutually complementary) occurs. If both inputs are then switched to 0 simultaneously, the device will enter an unpredictable or undefined state or a metastable state.

D Latch : This latch has only two inputs: *D* (data) and *En* (enable). The *D* input goes directly to the *S* input, and its complement is applied to the *R* input. As long as the enable input is at 0, the cross-coupled *SR* latch has both inputs at the 1 level and the circuit cannot change state regardless of the value of *D* . The *D* input is sampled when $En = 1$. If $D = 1$, the *Q* output goes to 1, placing the circuit in the set state. If $D = 0$, output *Q* goes to 0, placing the circuit in the reset state. The *D* latch receives that designation from its ability to hold *data* in its internal storage.

Master-Slave Flip-Flop,: Observe that the master changes when the pulse goes positive and the slave follows the change when the pulse goes negative.

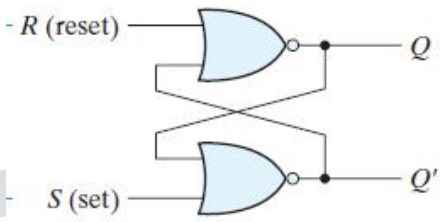
Edge-Triggered Flip-Flop: The construction of a *D* flip-flop with two *D* latches and an inverter is shown in Fig. The first latch is called the master and the second the slave. The circuit samples the *D* input and changes its

output Q only at the negative edge of the synchronizing or controlling clock (designated as *Clk*). When the clock is 0, the output of the inverter is 1. The slave latch is enabled, and its output Q is equal to the master output Y . The master latch is disabled because $Clk = 0$. When the input pulse changes to the logic-1 level, the data from the external D input are transferred to the master. The slave, however, is disabled as long as the clock remains at the 1 level, because its *enable* input is equal to 0. Any change in the input changes the master output at Y, but cannot affect the slave output. When the clock pulse returns to 0, the master is disabled and is isolated from the D input. At the same time, the slave is enabled and the value of Y is transferred to the output of the flip-flop at Q . Thus, *a change in the output of the flip-flop can be triggered only by and during the transition of the clock from 1* to 0. C

Flip-Flops(using IC 7476, and 7474): IC type 7476 consists of two *JK* master–slave flip-flops with preset and clear. The pin assignment for each flip-flop is shown in Fig IC type 7474 consists of two *D* positive-edge-triggered flip-flops with preset and clear. The pin assignment is shown in Fig.

Circuit

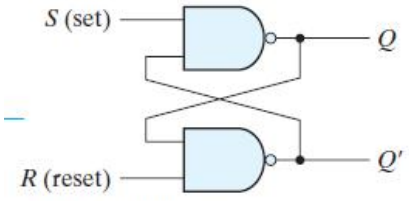
SR Latch



(a) Logic diagram

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

(b) Function table

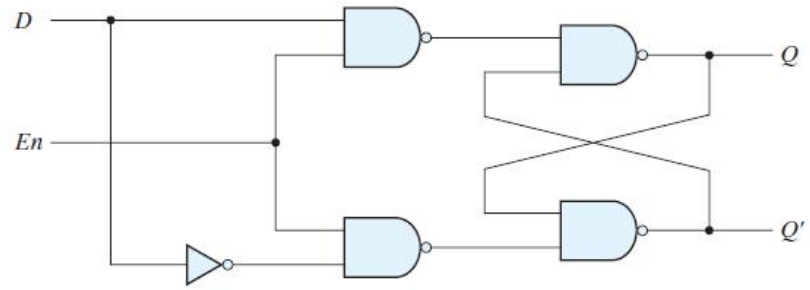


(a) Logic diagram

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)

(b) Function table

D Latch,

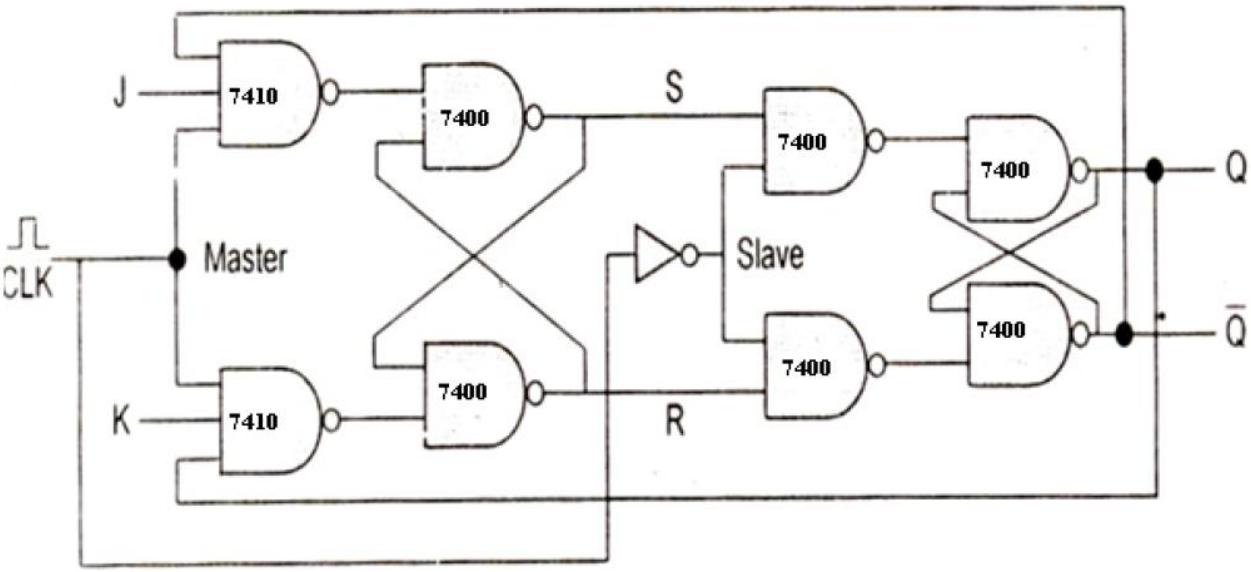


(a) Logic diagram

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

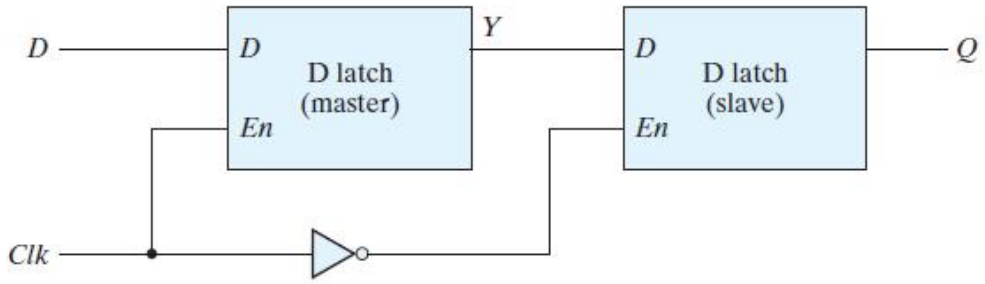
(b) Function table

Master-Slave Flip-Flop



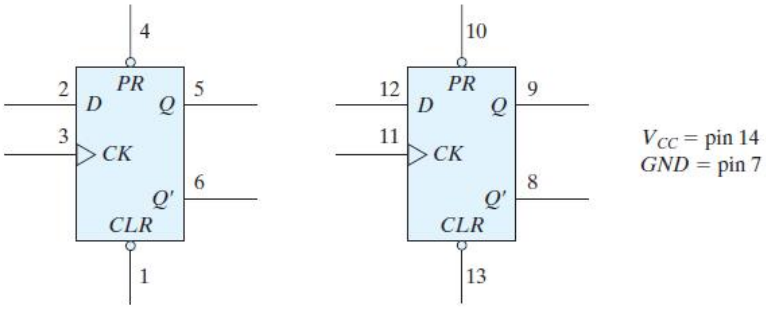
Inputs			Output
J	K	C	Q
0	0	⎍	Q_0
0	1	⎍	0
1	0	⎍	1
1	1	⎍	$\bar{Q}_0$

Edge-Triggered Flip-Flop



IC Flip-Flops(using IC 7476, and 7474)

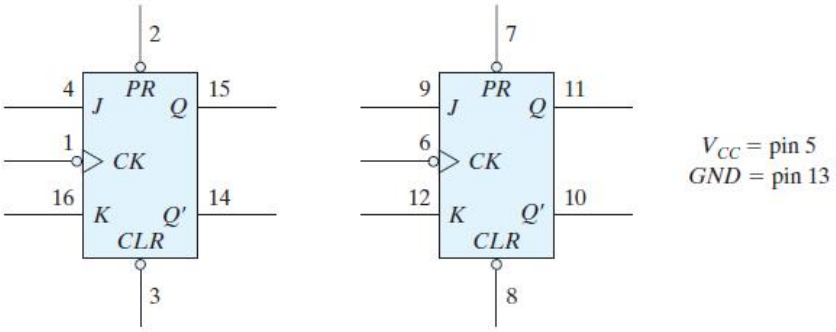
7474



Function table

Inputs				Outputs	
Preset	Clear	Clock	D	Q	Q'
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	1	1
1	1	↑	0	0	1
1	1	↑	1	1	0
1	1	0	X	No change	

7476



Function table

Inputs					Outputs	
Preset	Clear	Clock	J	K	Q	Q'
0	1	X	X	X	1	0
1	0	X	X	X	0	1
0	0	X	X	X	1	1
1	1	⊓	0	0	No change	
1	1	⊓	0	1	0	1
1	1	⊓	1	0	1	0
1	1	⊓	1	1	Toggle	

Procedure

1. Connections are made as per the circuit diagram
2. Switch on the power supply
3. Apply different combinations of inputs and observe the outputs; compare the outputs with the truth tables

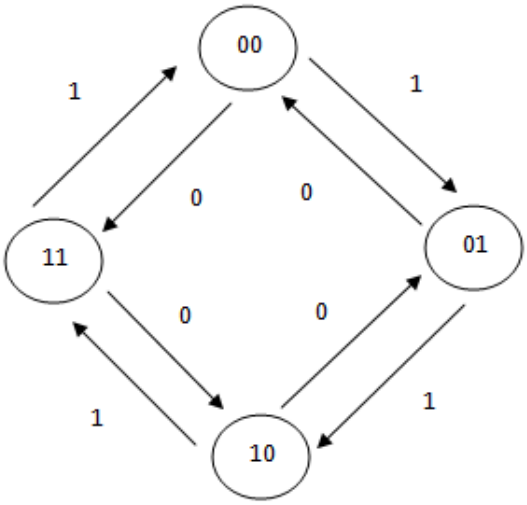
Result : Different logic Circuits are constructed and their truth tables are verified.

Exp.No…10……. Name of the Experiment: Up-Down counter with Enable
Aim: Sequential Circuits (Design, construct, and test Up-Down counter with Enable)
Objectives: To study the design, implementation and the working of Sequential Circuits.
Components: Bread board/ Kit , 7410, 7404, NTE754, 744075
Theory:

Design, construct, and test a two-bit counter that counts up or down. An enable input *E* determines whether the counter is on or off. If *E* = 0, the counter is disabled and remains at its present count even though clock pulses are applied to the flip-flops. If *E* = 1, the counter is enabled and a second input, *x,* determines the direction of the count. If *x* = 1, the circuit counts upward with the sequence 00, 01, 10, 11, and the count repeats. If *x* = 0, the circuit counts downward with the sequence 11, 10, 01, 00, and the count repeats. Do not use *E* to disable the clock. Design the sequential circuit with *E* and *x* as inputs.

Circuit

State digram



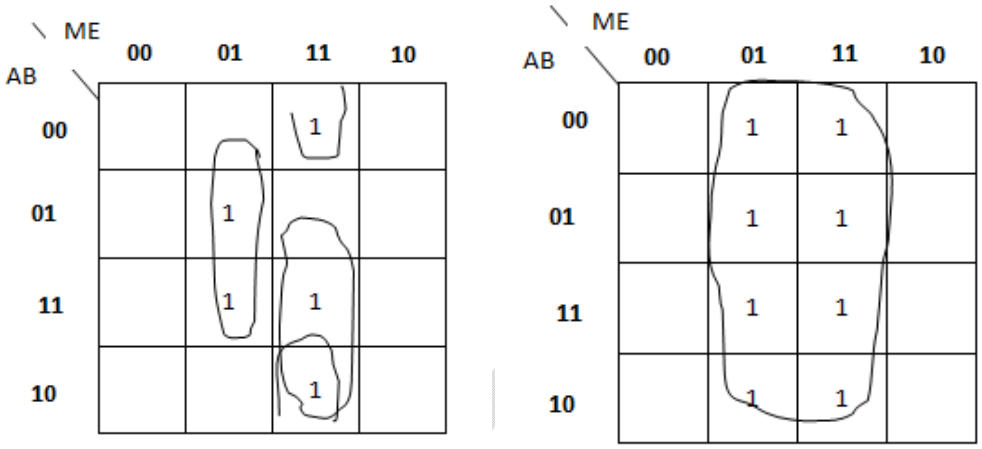
PS	NS	Required input
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

T ff Excitation Table

A	*B*	*M*	*E*	*OA*	*OB*	*TA*	*TB*
0	*0*	*0*	*0*	*0*	*0*	*0*	*0*
0	*0*	*0*	*1*	*0*	*1*	*0*	*1*
0	*0*	*1*	*0*	*0*	*0*	*0*	*0*
0	*0*	*1*	*1*	*1*	*1*	*1*	*1*
0	*1*	*0*	*0*	*0*	*1*	*0*	*0*
0	*1*	*0*	*1*	*1*	*0*	*1*	*1*
0	*1*	*1*	*0*	*0*	*1*	*0*	*0*
0	*1*	*1*	*1*	*0*	*0*	*0*	*1*
1	*0*	*0*	*0*	*1*	*0*	*0*	*0*
1	*0*	*0*	*1*	*1*	*1*	*0*	*1*
1	*0*	*1*	*0*	*1*	*0*	*0*	*0*
1	*0*	*1*	*1*	*0*	*1*	*1*	*1*
1	*1*	*0*	*0*	*1*	*1*	*0*	*0*
1	*1*	*0*	*1*	*0*	*0*	*1*	*1*
1	*1*	*1*	*0*	*1*	*1*	*0*	*0*
1	*1*	*1*	*1*	*1*	*0*	*1*	*1*

Note:- M=0= Up, M=1=Down, E=0=No Change, E=1=Change

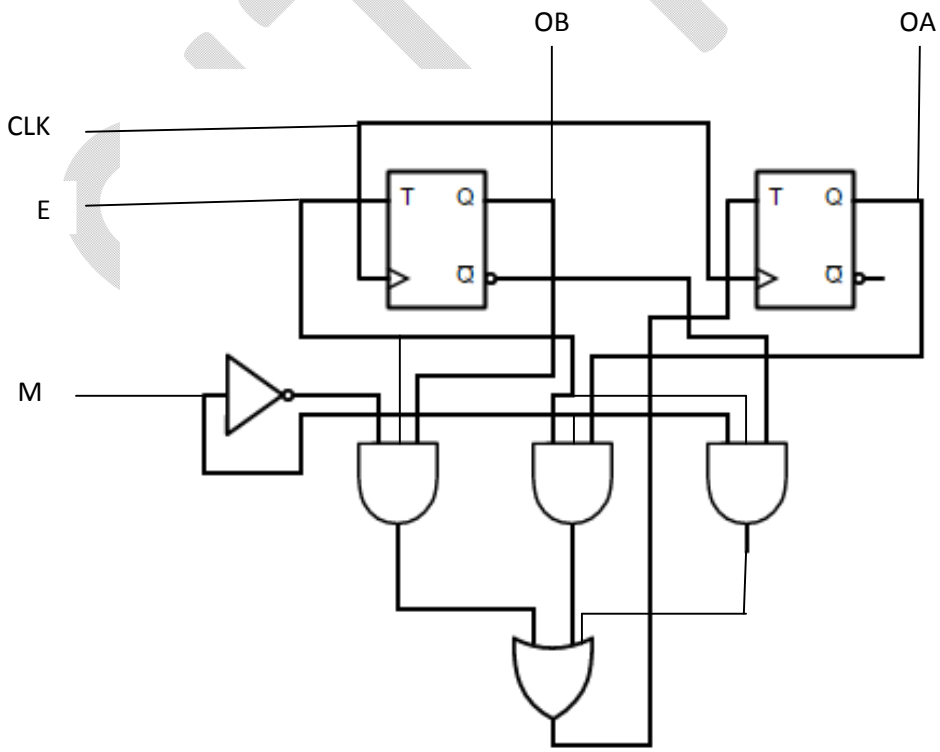
Truth table



TA=M'EB+AME+B'ME **TB=E**

Circuit

SR Latch



Procedure

1. Connections are made as per the circuit diagram
2. Switch on the power supply
3. Apply different combinations of inputs and observe the outputs; compare the outputs with the truth tables

Result : Different logic Circuits are constructed and their truth tables are verified.

Exp.No…11……. Name of the Experiment: Counter (User controlled counting pattern)

Aim: Design, construct, and test a counter that goes through a sequence of binary states (User controlled counting pattern) : 0, 3,5,6, and back to 0

Objectives: To study the design, construction , and working of counter with User controlled counting pattern

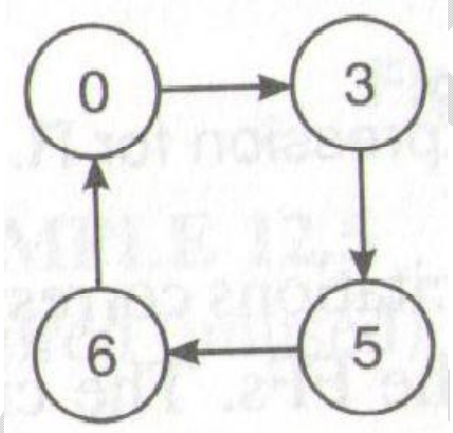
Components: Bread board/ Kit , NTE754

Theory:

Design, construct, and test a counter that goes through the following sequence of binary states: 0, 3,5,6, and back to 0 to repeat

Circuit

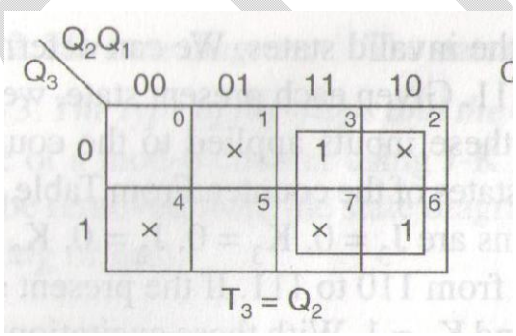
State digram

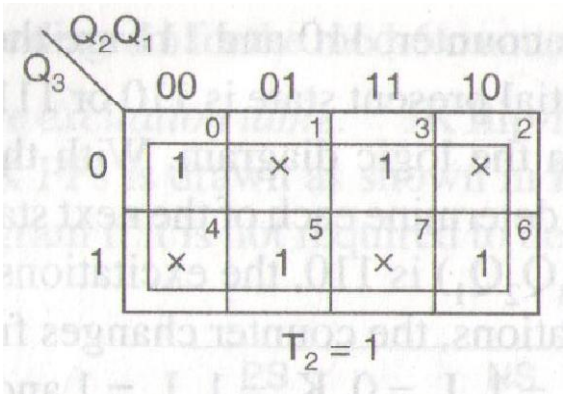


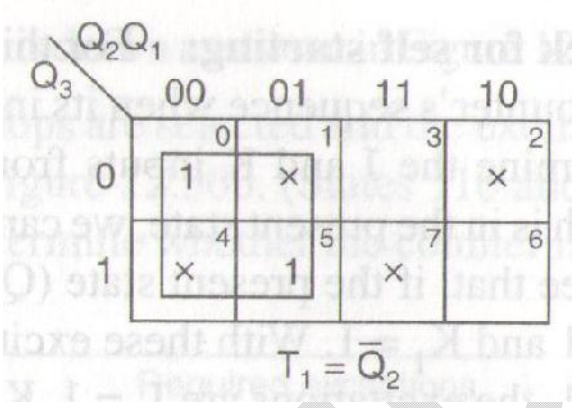
Excitation Table

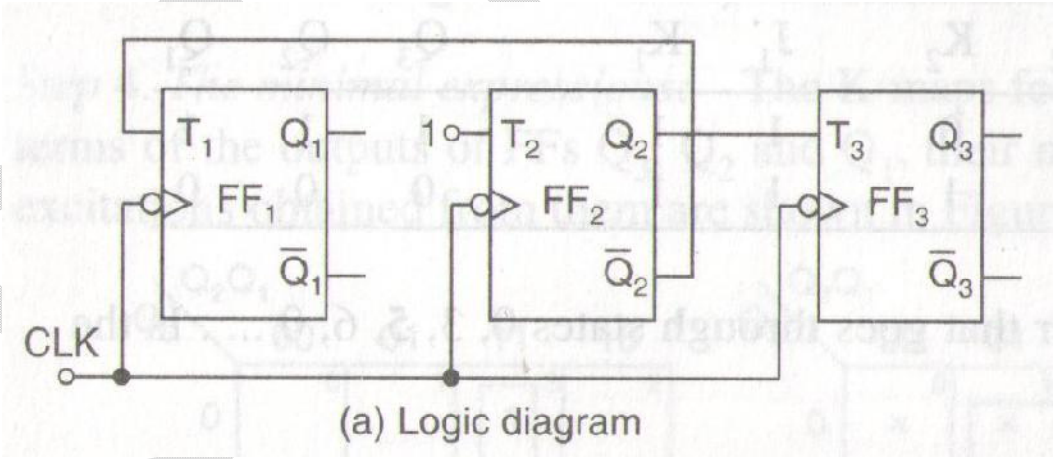
PS	NS	Required input
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

PS			NS			Required excitations		
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	T_3	T_2	T_1
0	0	0	0	1	1	0	1	1
0	1	1	1	0	1	1	1	0
1	0	1	1	1	0	0	1	1
1	1	0	0	0	0	1	1	0









(a) Logic diagram

Procedure

1. Connections are made as per the circuit diagram
2. Switch on the power supply
3. Apply different combinations of inputs and observe the outputs; compare the outputs with the truth tables

Result : Different logic Circuits are constructed and their truth tables are verified.

Exp.No…12……. Name of the Experiment: Ripple Counter, Synchronous counter, Decimal Counter

Aim: Construct, and test Ripple Counter, Synchronous counter, Decimal Counter

Objectives: To study the design , Construction , and the working of various Counters.

Components: Bread board/ Kit , 7476,7408

Theory: A binary ripple counter consists of series connection of complementing flip-flops with outputs of each flip-flop connected to the clock of the next higher order flip-flop. The flip-flop holding the least significant bit receives the incoming pulse. A complementing flip-flop can be obtained from a JK flip-flop with J & K inputs tied together or from a T flip-flop. The inputs of all flip-flops are connected to logic 1. This makes each flip-flop complement if the clock input goes through negative transition. IC 7476 consists of two JK flip-flops with PRESET & CLEAR. The pin diagram is as shown in figure.

Synchronous counter,

Construct a synchronous four-bit binary counter and check its operation. Use two 7476 ICs and one 7408 IC.

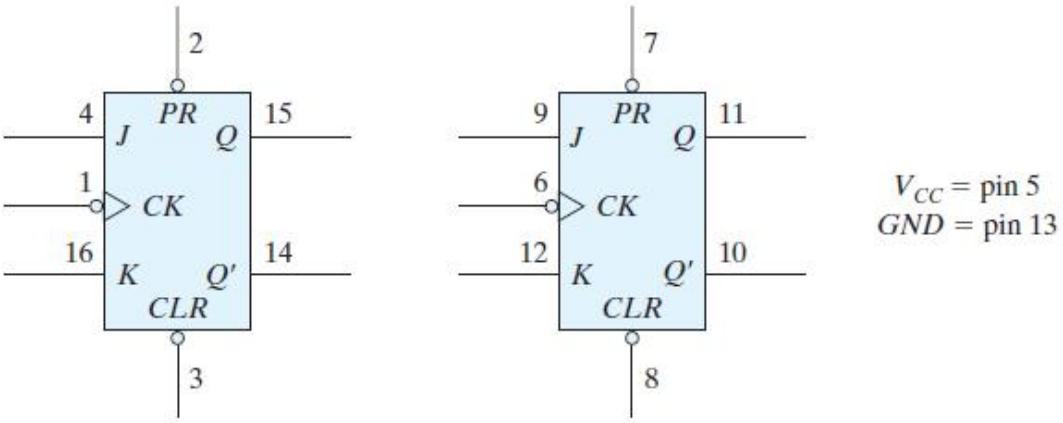
However, with the **Synchronous Counter**, the external clock signal is connected to the clock input of EVERY individual flip-flop within the Counter so that all of the flip-flops are clocked together simultaneously (in parallel) at the same time giving a fixed time relationship. In other words, changes in the output occur in "synchronisation" with the clock signal. The result of this synchronisation is that all the individual output bits changing state at exactly the same time in response to the common clock signal with no ripple effect and therefore, no propagation delay.

Decimal Counter

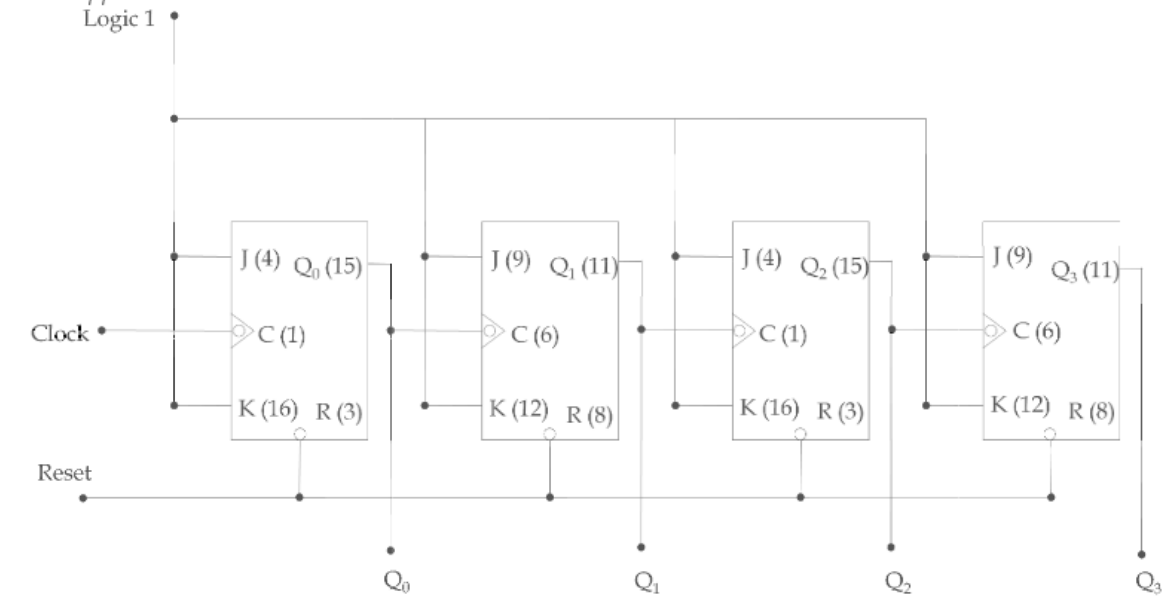
Design a synchronous BCD counter that counts from 0000 to 1001. Use two 7476 ICs and one 7408 IC. Test the counter for the proper sequence.

Circuit

State digram

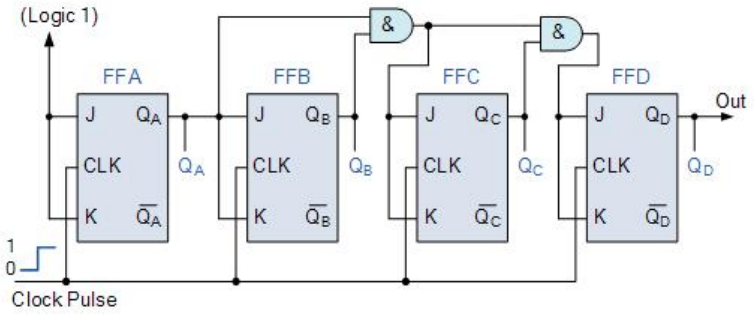


4-bit Ripple Counter:

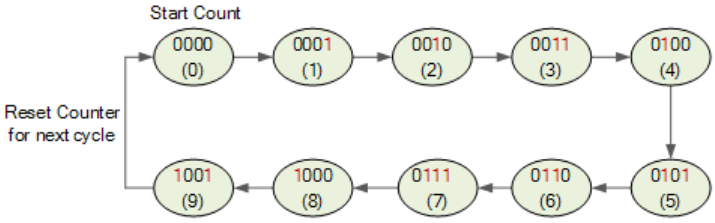


Synchronous counter,

Binary 4-bit Synchronous Up Counter



Decimal Counter

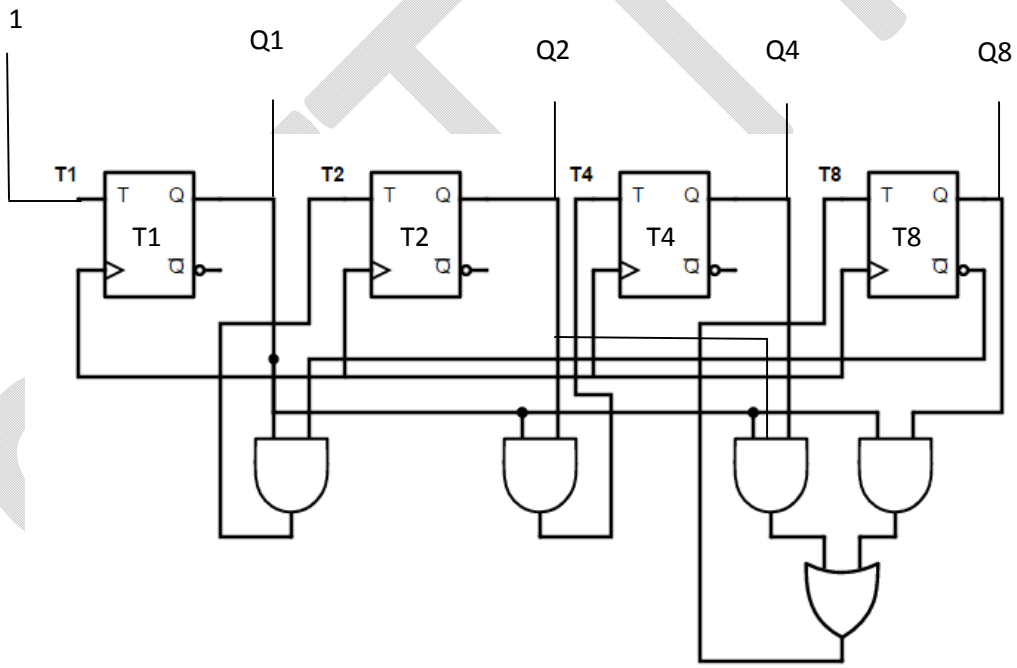


State Table for BCD Counter

Present State				Next State				Output	Flip-Flop Inputs			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	y	TQ_8	TQ_4	TQ_2	TQ_1
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	0	1
1	0	0	1	0	0	0	0	1	1	0	0	1

The flip-flop input equations can be simplified by means of maps. The unused states for minterms 10 to 15 are taken as don't-care terms. The simplified functions are

$$T_{Q1} = 1$$
$$T_{Q2} = Q'_8 Q_1$$
$$T_{Q4} = Q_2 Q_1$$
$$T_{Q8} = Q_8 Q_1 + Q_4 Q_2 Q_1$$



Procedure

1. Connections are made as per the circuit diagram
2. Switch on the power supply
3. Observe the outputs; compare the outputs with the truth tables

Result : Different logic Circuits are constructed and their truth tables are verified.

Exp.No…13……. Name of the Experiment: Binary counter with Parallel Load

Aim: Setup a Binary counter with Parallel Load (use IC 74161)

Objectives: To study the design, implementation and the working of a Binary counter with Parallel Load.

Components: Bread board/ Kit , 74161

Theory:

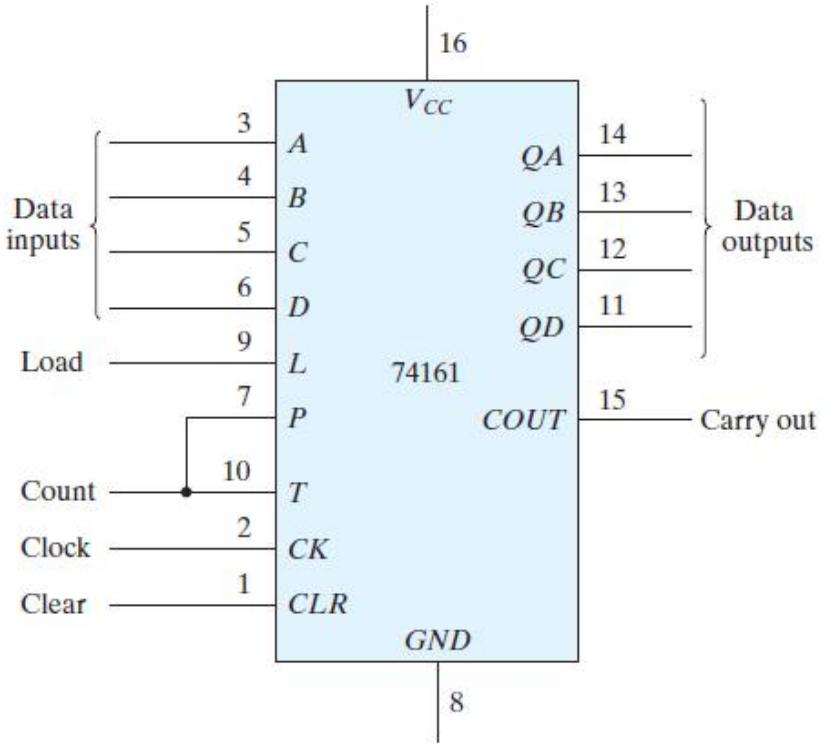
IC type 74161 is a four-bit synchronous binary counter with parallel load and asynchronous clear.. The pin assignments to the inputs and outputs are shown in Fig.. When the load signal is enabled, the four data inputs are transferred into four internal flip-flops, *QA* through *QD*, with *QD* being the most significant bit. There are two count-enable inputs called *P* and *T* . Both must be equal to 1 for the counter to operate. The function table is similar to Table 6.6, with one exception: The load input in the 74161 is enabled when equal to 0. To load the input data, the clear input must be equal to 1 and the load input must be equal to 0. The two count inputs have don't-care conditions and may be equal to either 1 or 0. The internal flip-flops trigger on the positive transition of the clock pulse. The circuit functions as a counter when the load input is equal to1 and both count inputs *P* and *T* are equal to 1. If either *P* or *T* goes to 0, the output

Circuit

State digram

Function table

Clear	Clock	Load	Count	Function
0	X	X	X	Clear outputs to 0
1	↑	0	X	Load input data
1	↑	1	1	Count to next binary value
1	↑	1	0	No change in output



Procedure

1. Connections are made as per the circuit diagram
2. Switch on the power supply
3. Apply different combinations of inputs and observe the outputs; compare the outputs with the truth tables

Result : Different logic Circuits are constructed and their truth tables are verified.

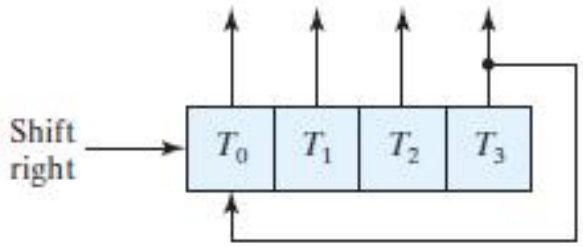
Exp.No…14……. Name of the Experiment: Shift Registers
Aim: Implement Shift Registers (Investigate the operation of Shift Registers, Ring Counter, Feedback Shift Register, Bidirectional Shift register, Bidirectional Shift Register with Parallel Load)
Objectives: To study the design and operations of Shift Registers
Components: Bread board/ Kit , 74195,74157
Theory:

Shift Registers,

A register capable of shifting the binary information held in each cell to its neighboring cell, in a selected direction, is called a *shift register*. The logical configuration of a shift register consists of a chain of flip-flops in cascade, with the output of one flip-flop connected to the input of the next flip-flop. All flip-flops receive common clock pulses, which activate the shift of data from one stage to the next.

Ring Counter,

A *ring counter* is a circular shift register with only one flip-flop being set at any particular time; all others are cleared. The single bit is shifted from one flip-flop to the next to produce the sequence of timing signals. Figure shows a four-bit shift register connected as a ring counter.

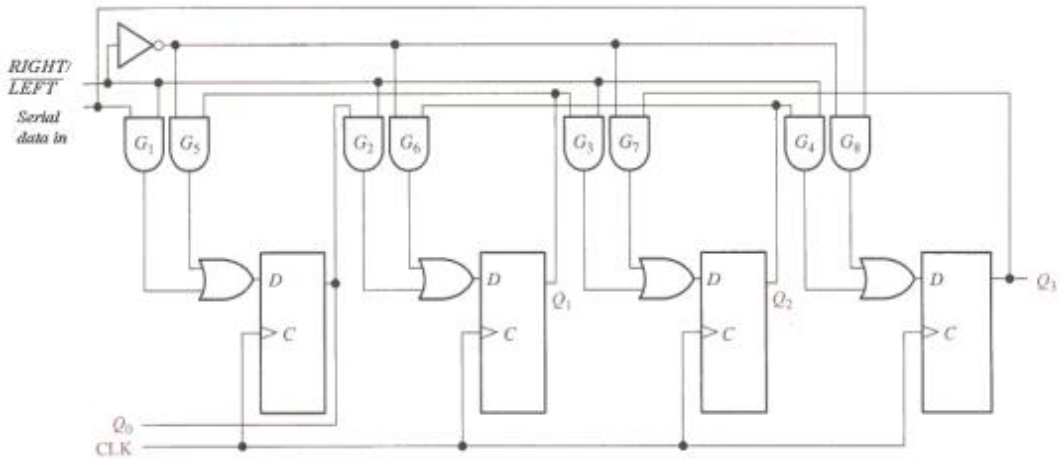


Feedback Shift Register,

A feedback shift register is a shift register whose serial input is connected to some function of selected register outputs a **linear-feedback shift register** (LFSR) is a shift register whose input bit is a linear function of its previous state. The most commonly used linear function of single bits is exclusive-or (XOR). Thus, an LFSR is most often a shift register whose input bit is driven by the XOR of some bits of the overall shift register value. The initial value of the LFSR is called the seed, and because the operation of the register is deterministic, the stream of values produced by the register is completely determined by its current (or previous) state. Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle. However, an LFSR with a well-chosen feedback function can produce a sequence of bits which appears random and which has a very long cycle.

Bidirectional Shift register,

A *bidirectional*, or *reversible*, shift register is one in which the data can be shift either left or right.



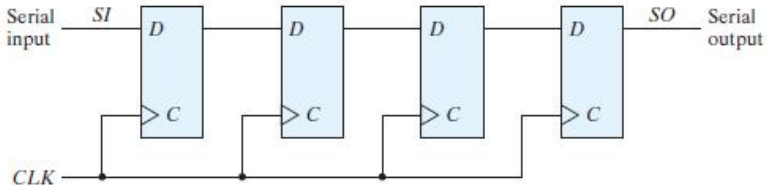
Bidirectional Shift Register with Parallel Load

Use the parallel-load condition to provide an initial value to the register, and connect the serial outputs to the serial inputs of both shifts in order not to lose the binary information while shifting.

Circuit

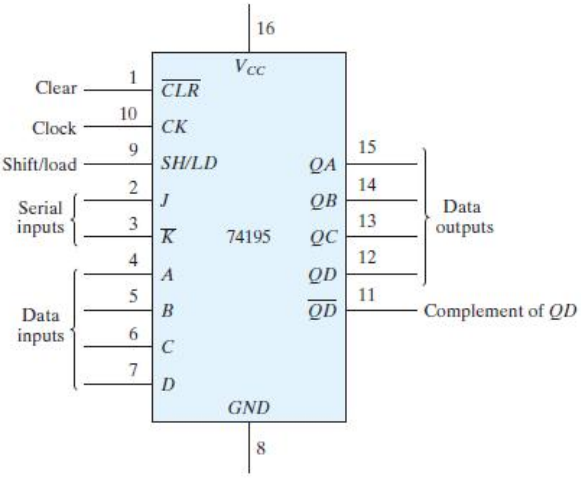
State digram

Shift Registers,



IC type 74195 is a four-bit shift register with parallel load and asynchronous clear. The pin assignments to the inputs and outputs are shown in Fig. . The single control line labeled *SH>LD* (shift/load) determines the synchronous operation of the register. When *SH>LD* = 0, the control input is in the load mode and the four data inputs are transferred into the four internal flip-flops, *QA* through *QD* . When *SH>LD* = 1, the control input is in the shift mode and the information in the register is shifted right from *QA* toward *QD* . The serial input into *QA* during the shift is determined from the *J* and *K* inputs. The two inputs behave like the *J* and the complement of *K* of a *JK* flip-flop. When both *J* and *K* are equal to 0, flip-flop

QA is cleared to 0 after the shift. If both inputs are equal to 1, *QA* is set to 1 after the shift. The other two conditions for the *J* and *K* inputs provide a complement or no change in the output of flip-flop *QA* after the shift. The function table for the 74195 shows the mode of operation of the register. When the clear input goes to 0, the four flip-flops clear to 0 asynchronously—that is, without the need of a clock. Synchronous operations are affected by a positive transition of the clock. To load the input data, *SH/LD* must be equal to 0 and a positive clock-pulse transition must occur. To shift right, *SH/LD* must be equal to 1. The *J* and *K* inputs must be connected together to form the serial input. Perform an experiment that will verify the operation of the 74195 IC. Show that it performs all the operations listed in the function table. Include in your function table the two conditions for *JK* = 01 and 10.



Function table

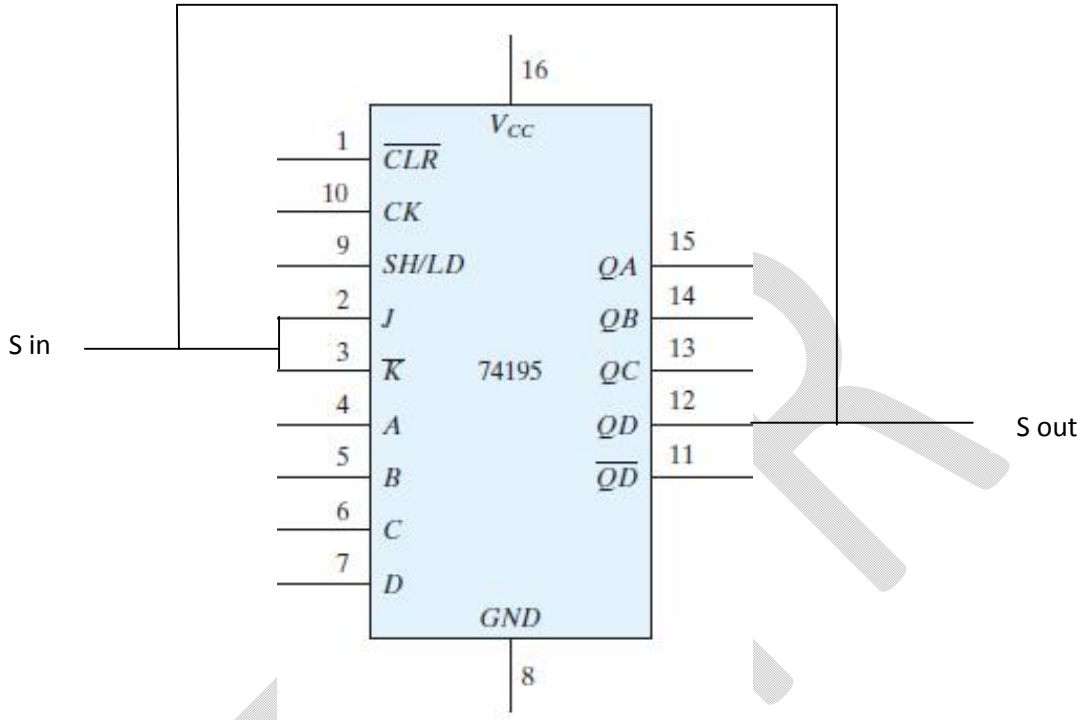
Clear	Shift/ load	Clock	*J*	$\overline{K}$	Serial input	Function
0	X	X	X	X	X	Asynchronous clear
1	X	0	X	X	X	No change in output
1	0	↑	X	X	X	Load input data
1	1	↑	0	0	0	Shift from *QA* toward *QD*, *QA* = 0
1	1	↑	1	1	1	Shift from *QA* toward *QD*, *QA* = 1

Ring Counter,

A ring counter is a circular shift register with the signal from the serial output *QD* going into the serial input. Connect the *J* and *K* input together to form the serial input. Use the load condition to preset the ring counter to an initial value of 1000. Rotate the single bit with the shift condition and check the state of the register after each clock pulse.

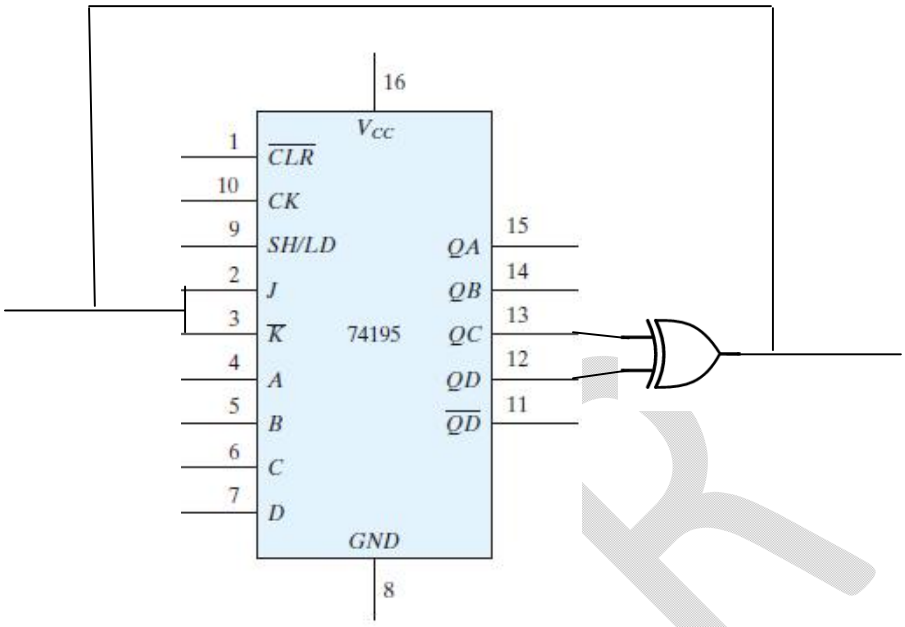
A switch-tail ring counter uses the complement output of *QD* for the serial input. Preset the switch-tail ring counter to 0000 and predict the sequence of states that

will result from shifting. Verify your prediction by observing the state sequence after each shift.



Feedback Shift Register,

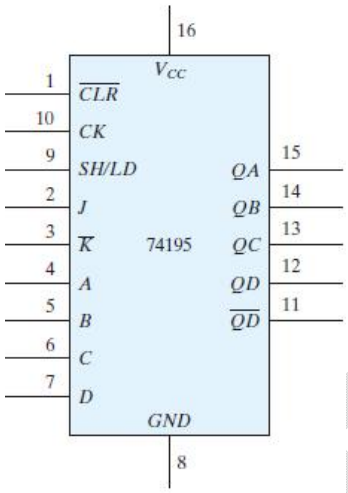
A feedback shift register is a shift register whose serial input is connected to some function of selected register outputs. Connect a feedback shift register whose serial input is the exclusive-OR of outputs *QC* and *QD* . Predict the sequence of states of the register, starting from state 1000. Verify your prediction by observing the state sequence after each clock pulse.



Bidirectional Shift register,

The 74195 IC can shift only right from *QA* toward *QD* . It is possible to convert the register to a bidirectional shift register by using the load mode to obtain a shift-left operation (from *QD* toward *QA*). This is accomplished by connecting the output of each flip-flop to the input of the flip-flop on its left and using the load mode of the *SH/LD* input as a shift-left control. Input *D* becomes the serial input for the shift left operation.

Connect the 74195 as a bidirectional shift register (without parallel load). Connect the serial input for shift right to a toggle switch. Construct the shift left as a ring counter by connecting the serial output *QA* to the serial input *D* . Clear the register and then check its operation by shifting a single 1 from the serial input switch. Shift right three more times and insert 0's from the serial input switch. Then rotate left with the shift-left (load) control. The single 1 should remain visible while shifting.

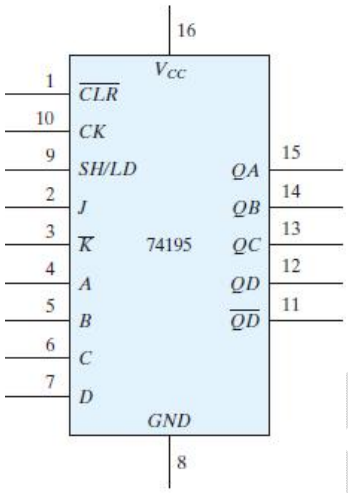


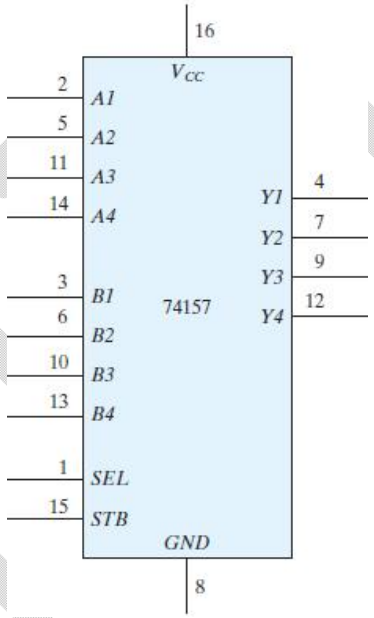
Bidirectional Shift Register with Parallel Load

The 74195 IC can be converted to a bidirectional shift register with parallel load in conjunction with a multiplexer circuit. We will use IC type 74157 for this purpose. The 74157 is a quadruple two-to-four-line multiplexer. The pin assignments to the inputs and outputs of the 74157 are shown in Fig. Note that the enable input is called a strobe in the 74157. Construct a bidirectional shift register with parallel load using the 74195 register and the 74157 multiplexer. The circuit should be able to perform the following operations:

1. Asynchronous clear
2. Shift right
3. Shift left
4. Parallel load
5. Synchronous clear

Derive a table for the five operations as a function of the clear, clock, and *SH/LD* inputs of the 74195 and the strobe and select inputs of the 74157. Connect the circuit and verify your function table. Use the parallel-load condition to provide an initial value to the register, and connect the serial outputs to the serial inputs of both shifts in order not to lose the binary information while shifting.





Procedure

1. Connections are made as per the circuit diagram
2. Switch on the power supply
3. Apply different combinations of inputs and observe the outputs; compare the outputs with the truth tables

Result : Different logic Circuits are constructed and their truth tables are verified.

COURSE TITLE : DIGITAL COMPUTER PRINCIPLES LAB
COURSE CODE : 3138
COURSE CATEGORY : B PERIODS/WEEK : 5
PERIODS/SEMESTER : 75 CREDITS : 3

Course General Outcomes:

The student should be able to do the experiments from the following topics on completion of corresponding
topic

1.0 To implement basic logic gates

1.0.1 Implement logic gates using basic components
1.0.2 Verify the Logic behaviour of various IC gates: 7408, 7432, 7404, 7400, 7402, 7486)
1.0.3 Implement basic gates using Universal Gates (NAND & NOR)
1.0.4 Simplification of Boolean Functions SOP & POS forms (Demonstrates the relationship between a Boolean Function and the corresponding logic diagram – using Map reduction method)
Eg: Plot the following Boolean function in a Map as well as implement in a logic diagram F= A'D + BD +B'C + AB'D

2.0 To design and Implement Combinational Circuits

2.0.1 Implement the following Combinational Circuits (Design, construct, and test combinational logic circuit that generates parity bit from four message bits)
2.0.2 Design a combinational circuit that converts a Gray code to binary, Decoder for a binary digit to BCD, and a seven segment indicator
2.0.3 Design, construct, and test a half-adder & a full-adder Implement using Basic gates and Universal gates
2.0.4 Implement a circuit using a four-bit binary parallel adder (IC 7483) implement AdderSubtractor, and a Magnitude Comparator)

3.0 To demonstrate synchronous sequential logic

3.0.1 Construct, Test, and investigate the operation of SR Latch, D Latch, Master-Slave Flip-Flop, Edge-Triggered Flip-Flop, IC Flip-Flops(using IC 7476, and 7474)
3.0.2 Sequential Circuits (Design, construct, and test Up-Down counter with Enable)
3.0.3 Design, construct, and test a counter that goes through a sequence of binary states (User controlled counting pattern)
3.0.4 Construct, and test Ripple Counter, Synchronous counter, Decimal Counter
3.0.5 Setup a Binary counter with Parallel Load (use IC 74161))
3.0.6 Study the operation of Shift Registers (Investigate the operation of Shift Registers, Ring Counter, Feedback Shift Register, Bidirectional Shift register, Bidirectional Shift Register with Parallel Load)