PET ENGINEERING COLLEGE, VALLIOOR

DEPARTMENT OF ECE

PRACTICAL RECORD

EC3352 DIGITAL SYSTEM DESIGN LABORATORY

BACHELOR OF ENGINEERING

In

ELECTRONICS AND COMMUNICATION ENGINEERING

ANNA UNIVERSITY CHENNAI

DEC 2023

PET ENGINEERING COLLEGE

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

An ISO 9001:2015 Certified Institution

Accredited by NAAC, Approved by AICTE, Recognized by Government of Tamil Nadu and Affiliated to Anna University

EC3352 DIGITAL SYSTEM DESIGN LABORATORY

COURSE CODE: C206 **SEM/YEAR: III/II**

S.NO	EXPERIMENT LIST
1.	Study of logic gates.
2.	Design and implementation of adders and subtractors using logic gates.
3.	Design and implementation of code converters using logic gates.
4.	Design and implementation of 4-bit binary adder/subtractor and BCD adder using IC 7483.
5.	Design and implementation of 2-bit magnitude comparator using logic gates, 8-bit magnitude comparator using IC 7485
6.	Design and implementation of 16-bit odd/even parity checker/ generator using IC 74180.
7.	Design and implementation of multiplexer and demultiplexer using logic gates and study of IC 74150 and IC 74154.
8.	Design and implementation of encoder and decoder using logic gates and study of IC 7445 and IC 74147.
9.	Construction and verification of 4-bit ripple counter and Mod-10/Mod-12 ripple counter.
10.	Design and implementation of 3-bit synchronous up/down counter.
11.	Implementation of SISO, SIPO, PISO and PIPO shift registers using flip- flops.

EC3352 - DIGITAL SYSTEM DESIGN LAB

LIST OF EXPERIMENTS

1. *Study of logic gates.*
2. *Design and implementation of adders and subtractors using logic gates.*
3. *Design and implementation of code converters using logic gates.*
4. *Design and implementation of 4-bit binary adder/subtractor and BCDadder using IC 7483.*
5. *Design and implementation of 2-bit magnitude comparator using logic gates, 8-bit magnitude comparator using IC 7485.*
6. *Design and implementation of 16-bit odd/even parity checker/ generator using IC 74180.*
7. *Design and implementation of multiplexer and demultiplexer using logic gates and study of IC 74150 and IC 74154.*
8. *Design and implementation of encoder and decoder using logic gates and study of IC 7445 and IC 74147.*
9. *Construction and verification of 4-bit ripple counter and Mod-10/Mod-12 ripple counter.*
10. *Design and implementation of 3-bit synchronous up/down counter.*
11. *Implementation of SISO, SIPO, PISO and PIPO shift registers using flip-flops.*

INDEX

EXP. NO	***DATE***	***NAME OF THE EXPERIMENT***	***PAGE NO***	***MARKS***	***SIGNATURE***

EXPT NO. :

STUDY OF LOGIC GATES

DATE :

AIM:

To study about logic gates and verify their truth tables.

APPARATUS REQUIRED:

SL No.	COMPONENT	SPECIFICATION	QTY
1.	AND GATE	IC 7408	1
2.	OR GATE	IC 7432	1
3.	NOT GATE	IC 7404	1
4.	NAND GATE 2 I/P	IC 7400	1
5.	NOR GATE	IC 7402	1
6.	X-OR GATE	IC 7486	1
7.	NAND GATE 3 I/P	IC 7410	1
8.	IC TRAINER KIT	-	1
9.	PATCH CORD	-	14

THEORY:

Circuit that takes the logical decision and the process are called logic gates. Each gate has one or more input and only one output.

OR, AND and NOT are basic gates. NAND, NOR and X-OR are known as universal gates. Basic gates form these gates.

AND GATE:

The AND gate performs a logical multiplication commonly known as AND function. The output is high when both the inputs are high. The output is low level when any one of the inputs is low.

OR GATE:

The OR gate performs a logical addition commonly known as OR function. The output is high when any one of the inputs is high. The output is low level when both the inputs are low.

NOT GATE:

The NOT gate is called an inverter. The output is high when the input is low. The output is low when the input is high.

NAND GATE:

The NAND gate is a contraction of AND-NOT. The output is high when both inputs are low and any one of the input is low .The output is low level when both inputs are high.

NOR GATE:

The NOR gate is a contraction of OR-NOT. The output is high when both inputs are low. The output is low when one or both inputs are high.

X- OR GATE:

The output is high when any one of the inputs is high. The output is low when both the inputs are low and both the inputs are high.

PROCEDURE:

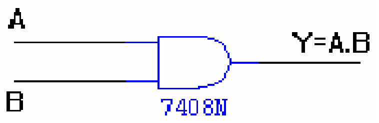
(i) Connections are given as per circuit diagram.

(ii) Logical inputs are given as per circuit diagram.

(iii) Observe the output and verify the truth table.

AND GATE:

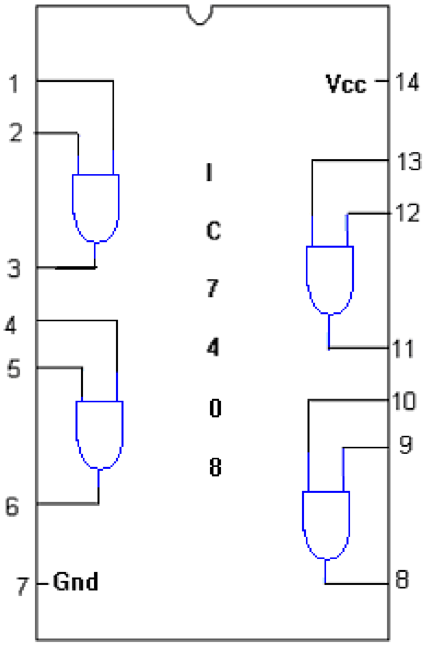
SYMBOL:



TRUTH TABLE

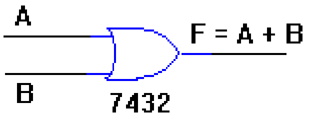
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

PIN DIAGRAM:



OR GATE:

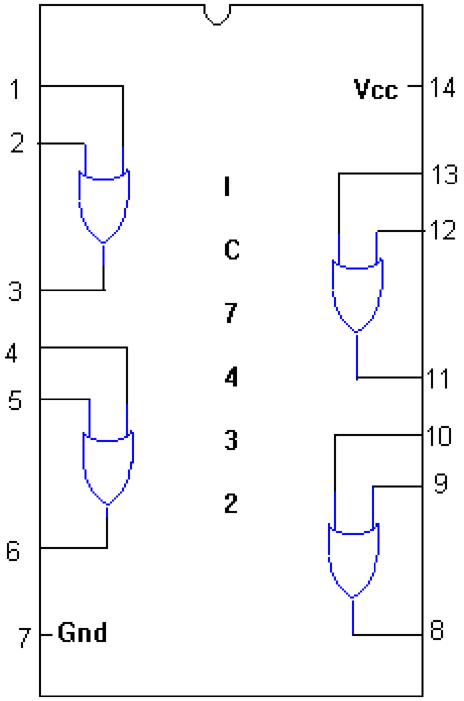
SYMBOL :



TRUTH TABLE

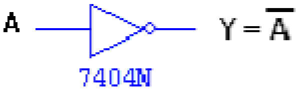
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

PIN DIAGRAM :



NOT GATE:

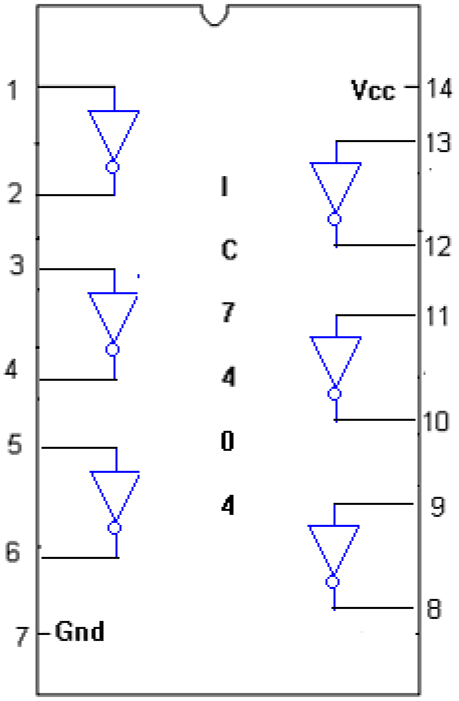
SYMBOL:



TRUTH TABLE :

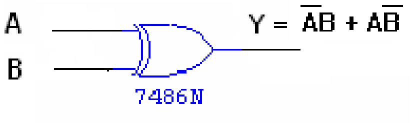
A	$\overline{A}$
0	1
1	0

PIN DIAGRAM:



X-OR GATE :

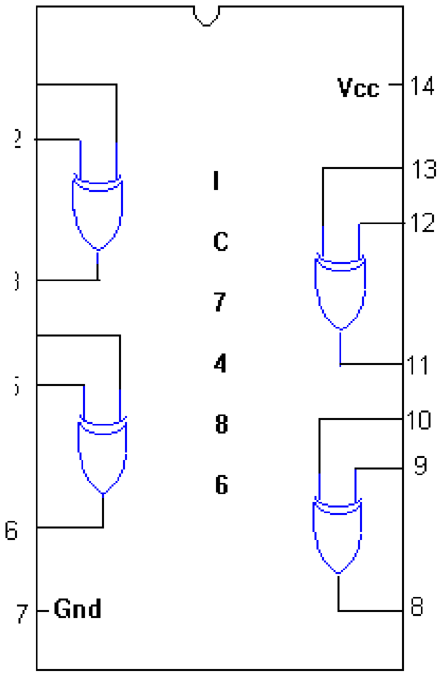
SYMBOL :



TRUTH TABLE :

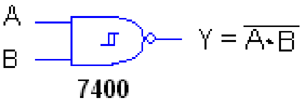
A	B	$\bar{A}B + A\bar{B}$
0	0	0
0	1	1
1	0	1
1	1	0

PIN DIAGRAM :



2- INPUT NAND GATE:

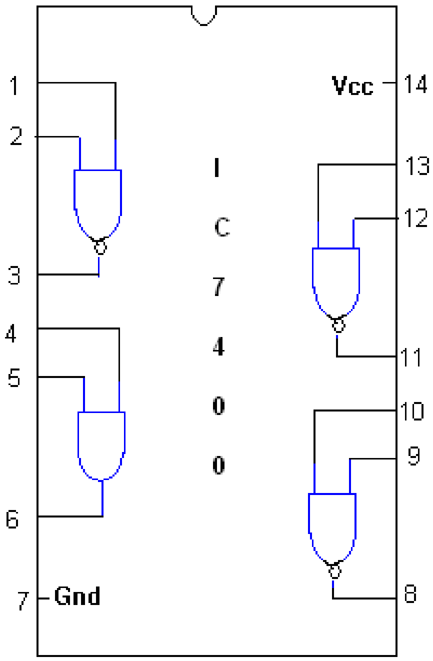
SYMBOL:



TRUTH TABLE

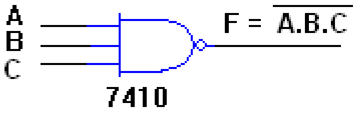
A	B	$\overline{A\cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

PIN DIAGRAM:

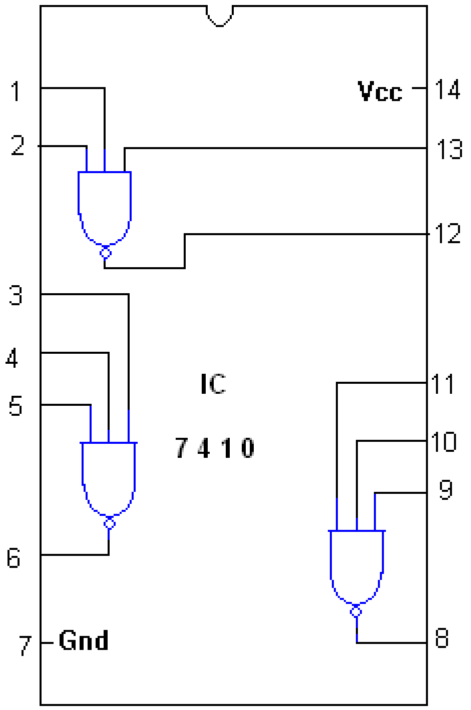


3- INPUT NAND GATE :

SYMBOL :



PIN DIAGRAM :

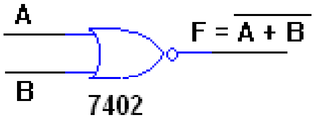


TRUTH TABLE

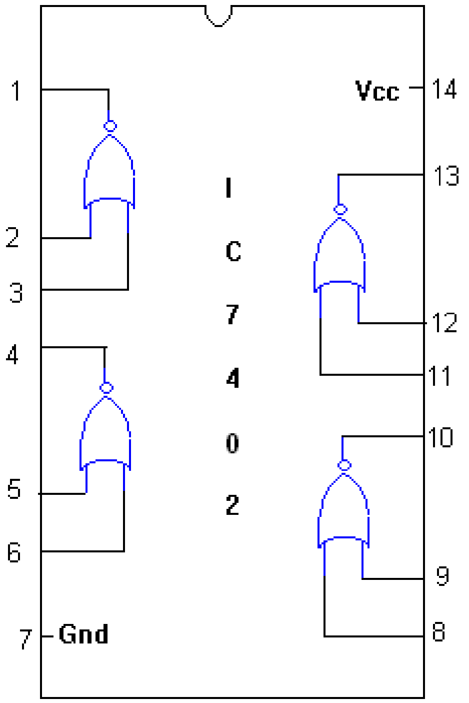
A	B	C	$\overline{A.B.C}$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

NOR GATE:

SYMBOL :



PIN DIAGRAM :



TRUTH TABLE

A	B	$\overline{A+B}$
0	0	1
0	1	1
1	0	1
1	1	0

RESULT:

EXPT NO. : <u>DESIGN OF ADDER AND SUBTRACTOR</u>

DATE :

AIM:

To design and construct half adder, full adder, half subtractor and full subtractor circuits and verify the truth table using logic gates.

APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	AND GATE	IC 7408	1
2.	X-OR GATE	IC 7486	1
3.	NOT GATE	IC 7404	1
4.	OR GATE	IC 7432	1
3.	IC TRAINER KIT	-	1
4.	PATCH CORDS	-	23

THEORY:

HALF ADDER:

A half adder has two inputs for the two bits to be added and two outputs one from the sum ' S' and other from the carry ' c' into the higher adder position. Above circuit is called as a carry signal from the addition of the less significant bits sum from the X-OR Gate the carry out from the AND gate.

FULL ADDER:

A full adder is a combinational circuit that forms the arithmetic sum of input; it consists of three inputs and two outputs. A full adder is useful to add three bits at a time but a half adder cannot do so. In full adder sum output will be taken from X-OR Gate, carry output will be taken from OR Gate.

HALF SUBTRACTOR:

The half subtractor is constructed using X-OR and AND Gate. The half subtractor has two input and two outputs. The outputs are difference and borrow. The difference can be applied using X-OR Gate, borrow output can be implemented using an AND Gate and an inverter.

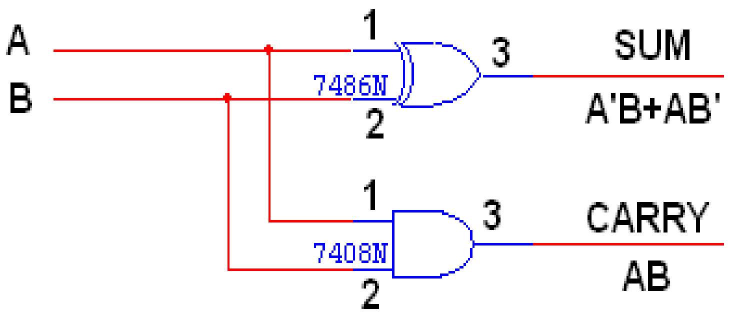
FULL SUBTRACTOR:

The full subtractor is a combination of X-OR, AND, OR, NOT Gates. In a full subtractor the logic circuit should have three inputs and two outputs. The two half subtractor put together gives a full subtractor .The first half subtractor will be C and A B. The output will be difference output of full subtractor. The expression AB assembles the borrow output of the half subtractor and the second term is the inverted difference output of first X-OR.

LOGIC

DIAGRAM:

HALF ADDER



TRUTH TABLE:

A	*B*	*CARRY*	*SUM*
0	*0*	*0*	*0*
0	*1*	*0*	*1*

1	*0*	*0*	*1*
1	*1*	*1*	*0*

K-Map for SUM:

A \ B	00	01
00		1
01	1	

K-Map for CARRY:

A \ B	00	01
00		
01		1

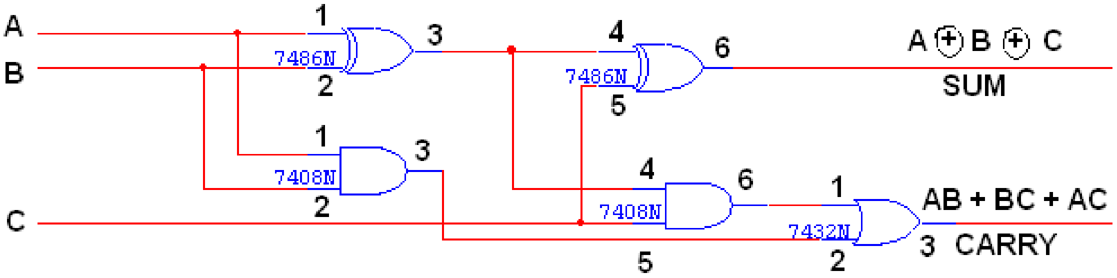
SUM = A'B + AB'

CARRY = AB

LOGIC DIAGRAM:

FULL ADDER

FULL ADDER USING TWO HALF ADDER

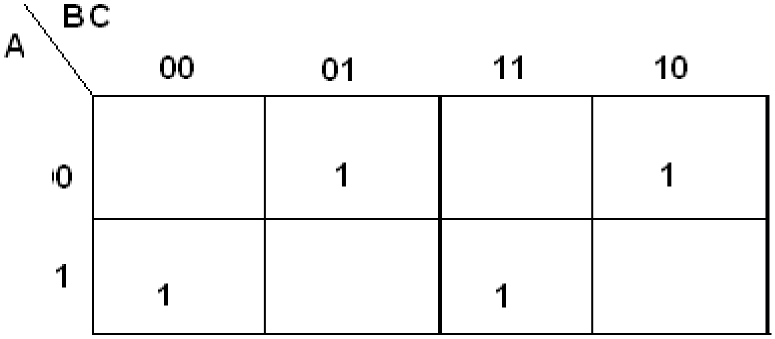


TRUTH TABLE:

A	*B*	*C*	*CARRY*	*SUM*

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*

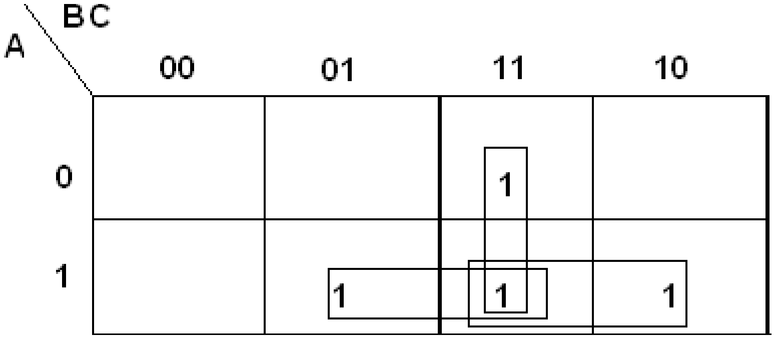
K-Map for SUM:



A \ BC	00	01	11	10
0		1		1
1	1		1	

SUM = A'B'C + A'BC' + ABC' + ABC

K-Map for CARRY:

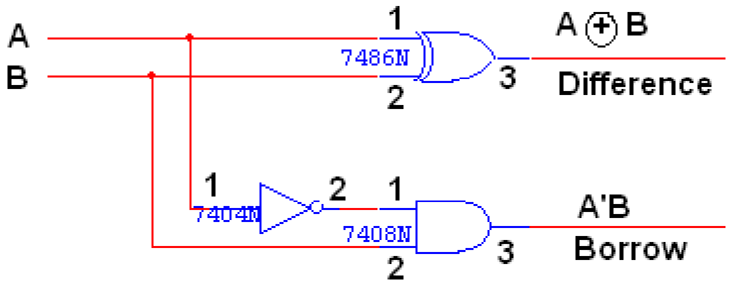


A \ BC	00	01	11	10
0			1	
1		1	1	1

CARRY = AB + BC

+ ACLOGIC DIAGRAM:

HALF SUBTRACTOR



TRUTH TABLE:

A	B	BORROW	DIFFERENCE
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

K-Map for DIFFERENCE:

A \ B	00	01
00		1
01	1	

DIFFERENCE = A’B + AB’

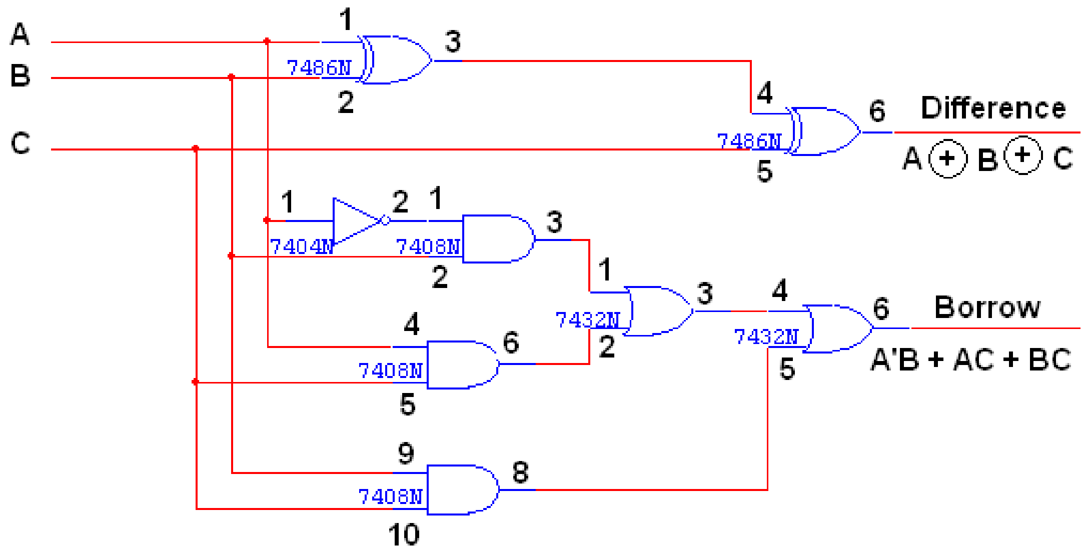
K-Map for BORROW:

A \ B	00	01
00		1
01		

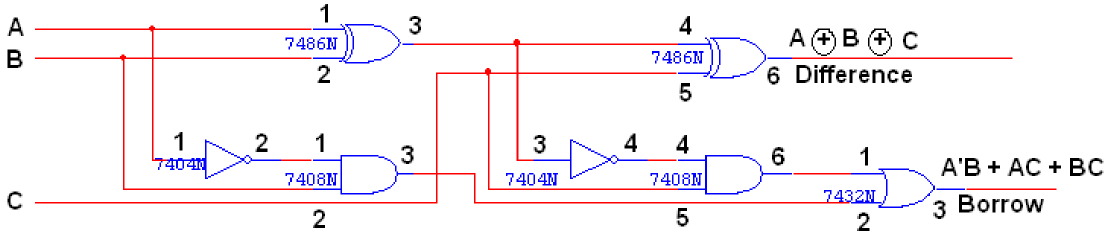
BORROW = A'B

LOGIC DIAGRAM:

<u>FULL SUBTRACTOR</u>



FULL SUBTRACTOR USING TWO HALF SUBTRACTOR:

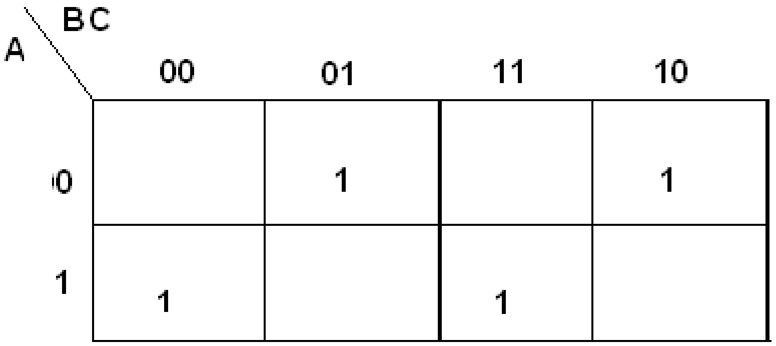


TRUTH TABLE:

A	*B*	*C*	*BORROW*	*DIFFERENCE*
0	*0*	*0*	*0*	*0*

0	*0*	*1*	*1*	*1*
0	*1*	*0*	*1*	*1*
0	*1*	*1*	*1*	*0*
1	*0*	*0*	*0*	*1*
1	*0*	*1*	*0*	*0*
1	*1*	*0*	*0*	*0*
1	*1*	*1*	*1*	*1*

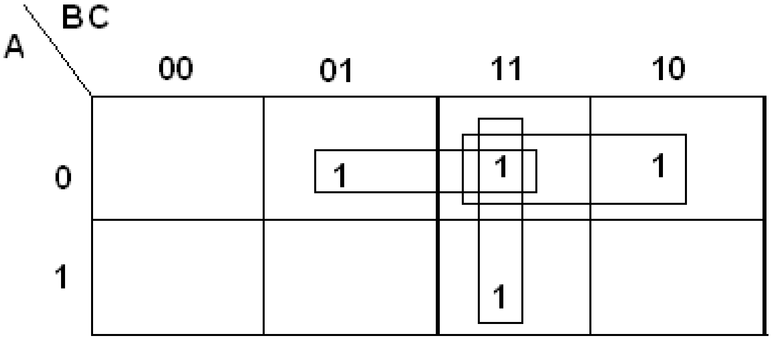
K-Map for Difference:



A \ BC	00	01	11	10
0		1		1
1	1		1	

Difference = A’B’C + A’BC’ + AB’C’ + ABC

K-Map for Borrow:



A \ BC	00	01	11	10
0		1	1	1
1			1	

Borrow = A’B + BC + A’C

PROCEEDURE:

(i) Connections are given as per circuit diagram.

(ii) Logical inputs are given as per circuit diagram.

(iii) Observe the output and verify the truth table.

RESULT:

EXPT NO.:

DATE :

DESIGN AND IMPLEMENTATION OF CODE CONVERTOR

AIM : *To design and implement 4-bit*

(i) Binary to gray code converter
(ii) Gray to binary code converter
(iii) BCD to excess-3 code converter
(iv) Excess-3 to BCD code converter

APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	X-OR GATE	IC 7486	1

2.	AND GATE	IC 7408	1
3.	OR GATE	IC 7432	1
4.	NOT GATE	IC 7404	1
5.	IC TRAINER KIT	-	1
6.	PATCH CORDS	-	35

THEORY:

The availability of large variety of codes for the same discrete elements of information results in the use of different codes by different systems. A conversion circuit must be inserted between the two systems if each uses different codes for same information. Thus, code converter is a circuit that makes the two systems compatible even though each uses different binary code.

The bit combination assigned to binary code to gray code. Since each code uses four bits to represent a decimal digit. There are four inputs and four outputs. Gray code is a non-weighted code.

The input variable are designated as B3, B2, B1, B0 and the output variables are designated as C3, C2, C1, Co. from the truth table, combinational circuit is designed. The Boolean functions are obtained from K-Map for each output variable.

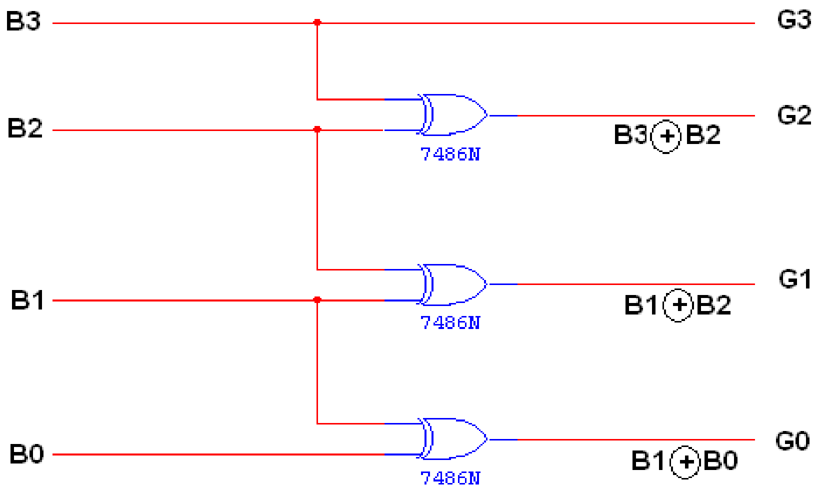
A code converter is a circuit that makes the two systems compatible even though each uses a different binary code. To convert from binary code to Excess-3 code, the input lines must supply the bit combination of elements as specified by code and the output lines generate the corresponding bit combination of code. Each one of the four maps represents one of the four outputs of the circuit as a function of the four input variables.

A two-level logic diagram may be obtained directly from the

Boolean expressions derived by the maps. These are various other possibilities for a logic diagram that implements this circuit. Now the OR gate whose output is C+D has been used to implement partially each of three outputs.

LOGIC DIAGRAM:

<u>BINARY TO GRAY CODE CONVERTOR</u>

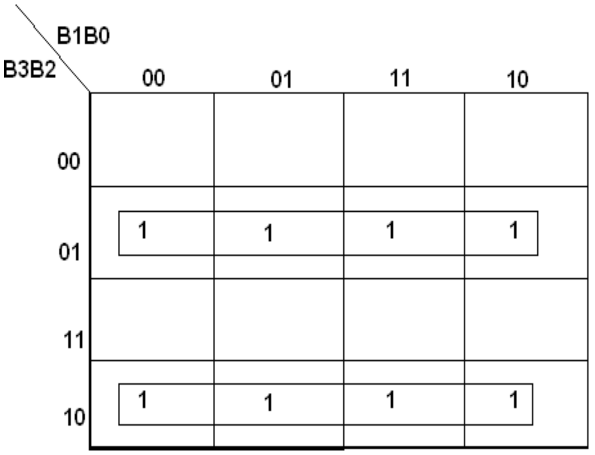


K-Map for G_3:

B3B2 \ B1B0	00	01	11	10
00				
01				
11	1	1	1	1
10	1	1	1	1

$$G_3 = B_3$$

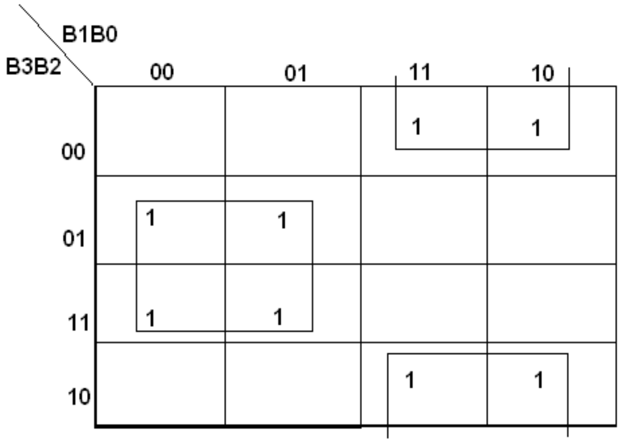
K-Map for G_2:



B1B0 B3B2	00	01	11	10
00				
01	1	1	1	1
11				
10	1	1	1	1

$G2 = B3 \oplus B2$

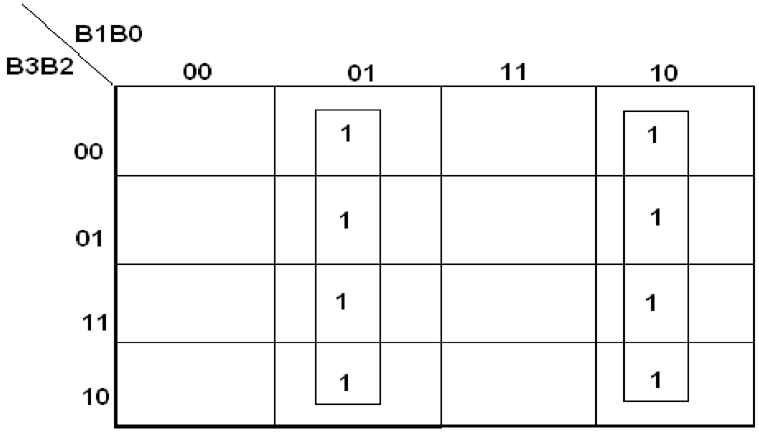
K-Map for G_1:



B1B0 B3B2	00	01	11	10
00			1	1
01	1	1		
11	1	1		
10			1	1

$G1 = B1 \oplus B2$

K-Map for G_0:



B1B0 B3B2	00	01	11	10
00		1		1
01		1		1
11		1		1
10		1		1

$$G0 = B1 \oplus B0$$

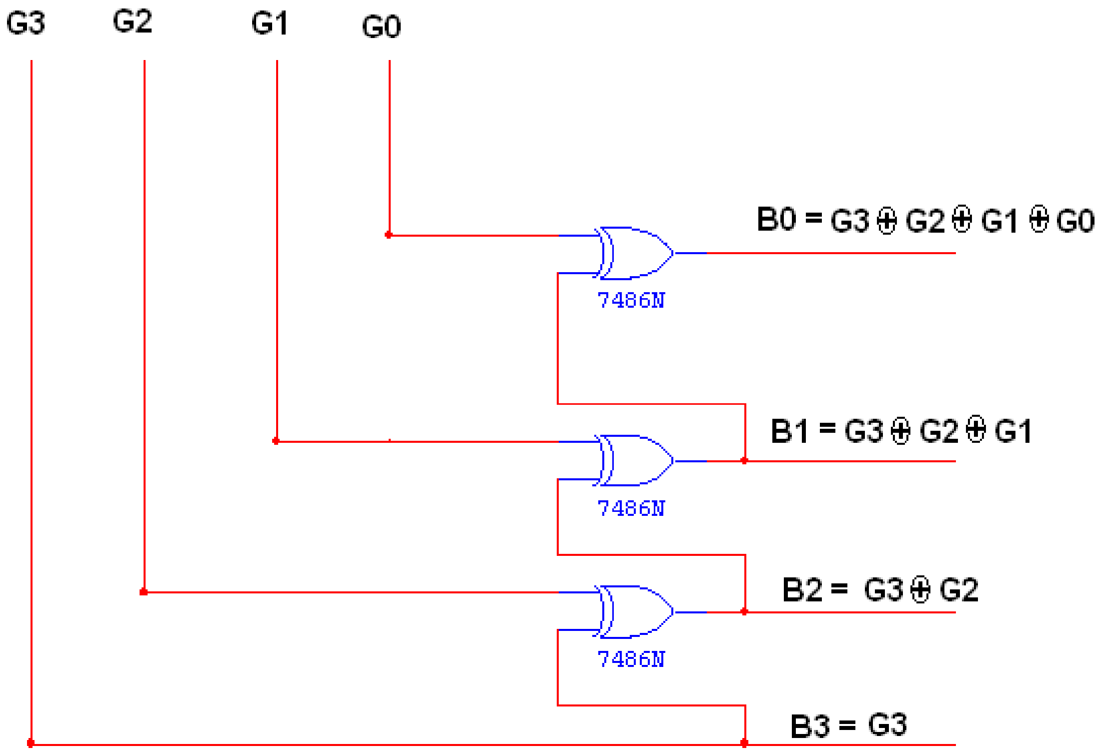
TRUTH TABLE:

| Binary input | Gray code output |

B3	B2	B1	B0	G3	G2	G1	G0
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

LOGIC DIAGRAM:

GRAY CODE TO BINARY CONVERTOR

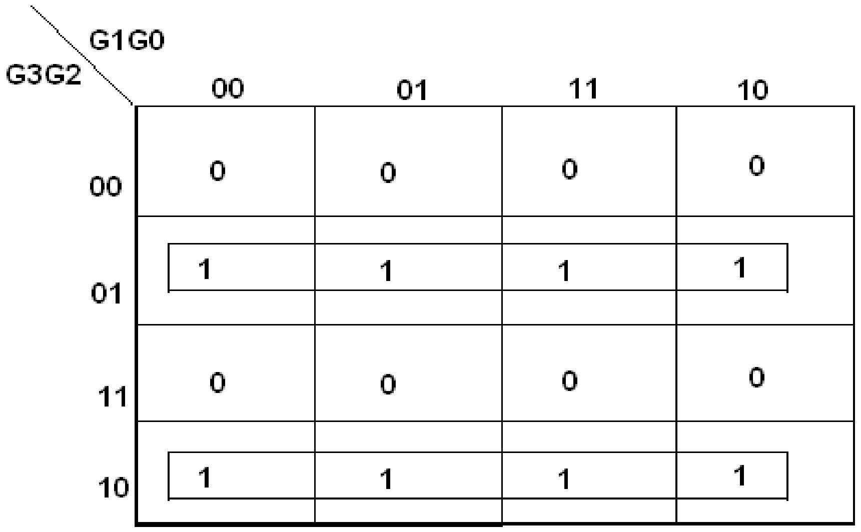


K-Map for B_3:

G3G2 \ G1G0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

B3 = G3

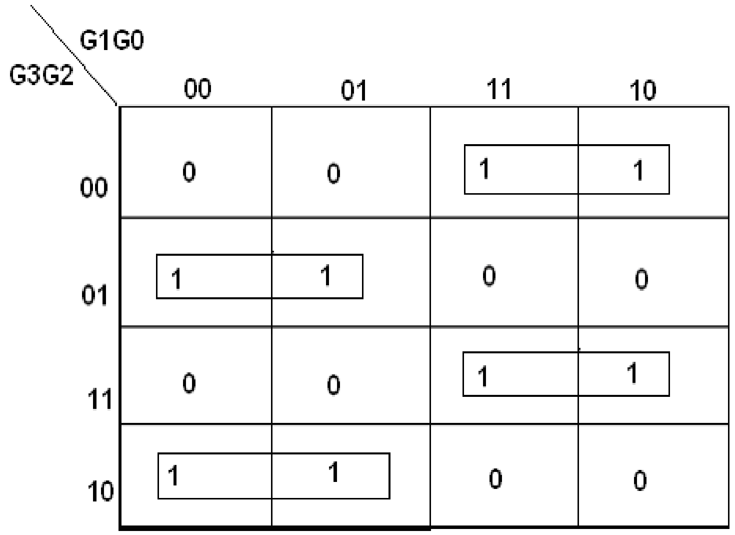
K-Map for B_2:



G3G2 \ G1G0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

B2 = G3⊕G2

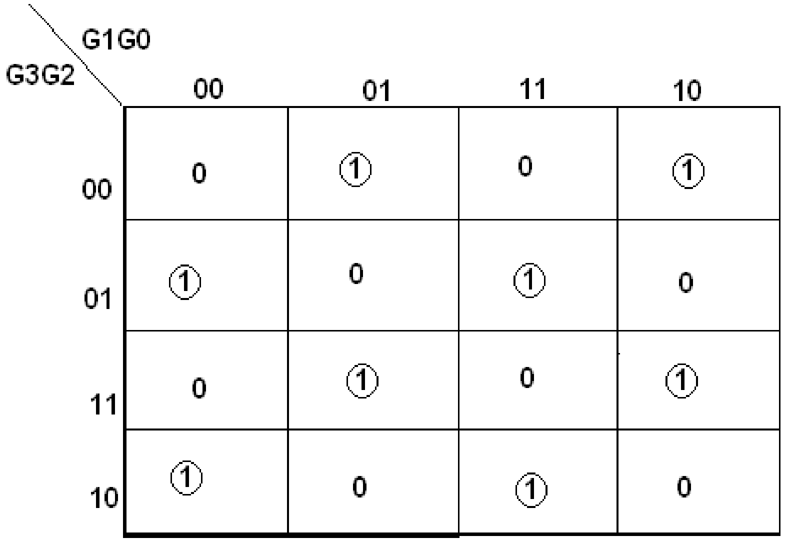
K-Map for B_1:



G3G2 \ G1G0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	0	0	1	1
10	1	1	0	0

B1 = G3⊕G2⊕G1

K-Map for B₀:



G3G2 \ G1G0	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

$$B0 = G3 \oplus G2 \oplus G1 \oplus G0$$

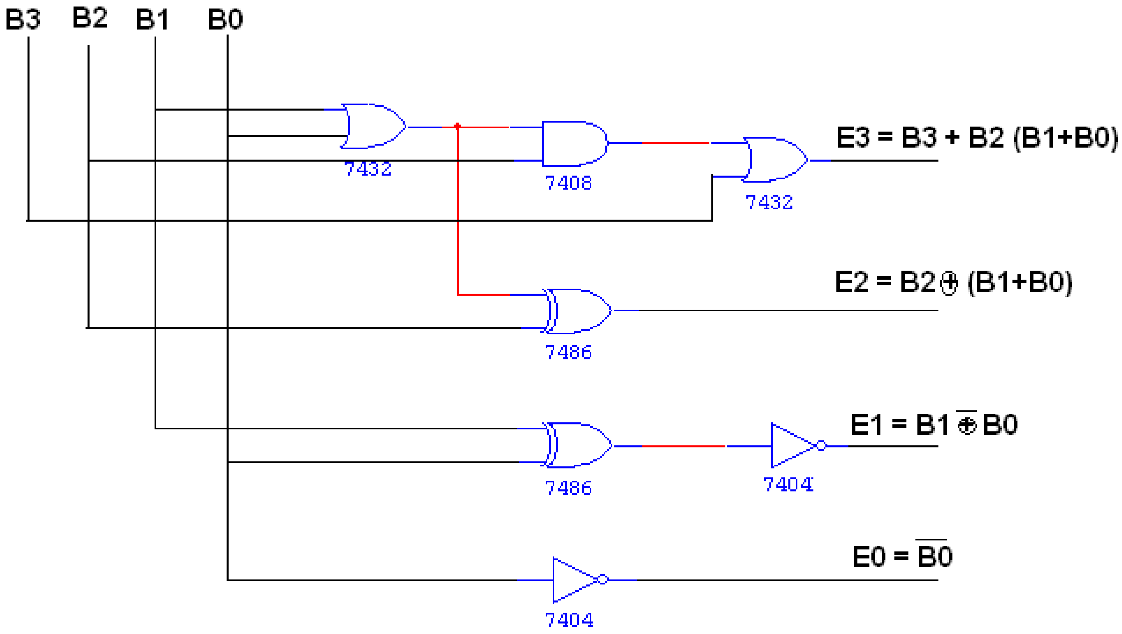
TRUTH TABLE:

| Gray Code | Binary Code |

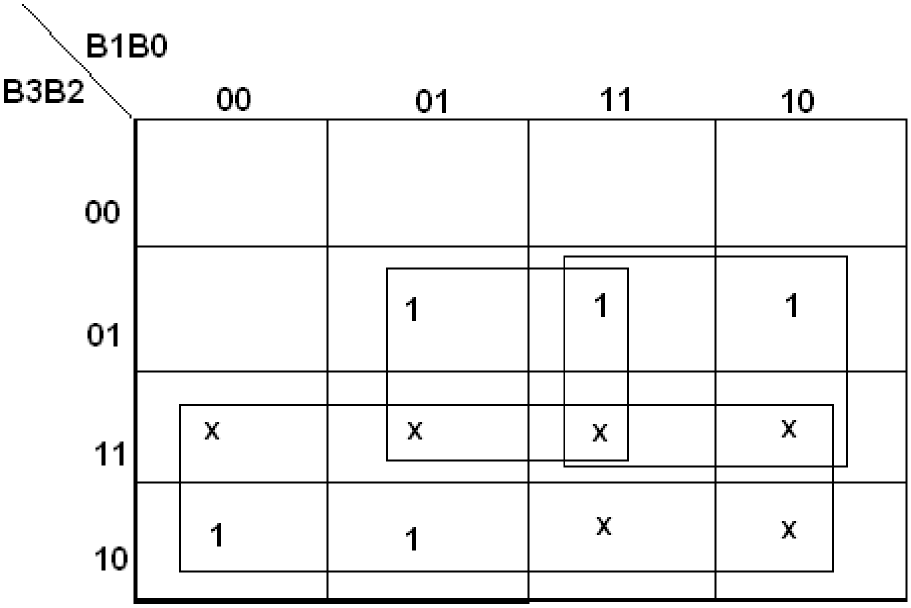
G3	*G2*	*G1*	*G0*	*B3*	*B2*	*B1*	*B0*
0	*0*	*0*	*0*	*0*	*0*	*0*	*0*
0	*0*	*0*	*1*	*0*	*0*	*0*	*1*
0	*0*	*1*	*1*	*0*	*0*	*1*	*0*
0	*0*	*1*	*0*	*0*	*0*	*1*	*1*
0	*1*	*1*	*0*	*0*	*1*	*0*	*0*
0	*1*	*1*	*1*	*0*	*1*	*0*	*1*
0	*1*	*0*	*1*	*0*	*1*	*1*	*0*
0	*1*	*0*	*0*	*0*	*1*	*1*	*1*
1	*1*	*0*	*0*	*1*	*0*	*0*	*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*

LOGIC DIAGRAM:

BCD TO EXCESS-3 CONVERTOR



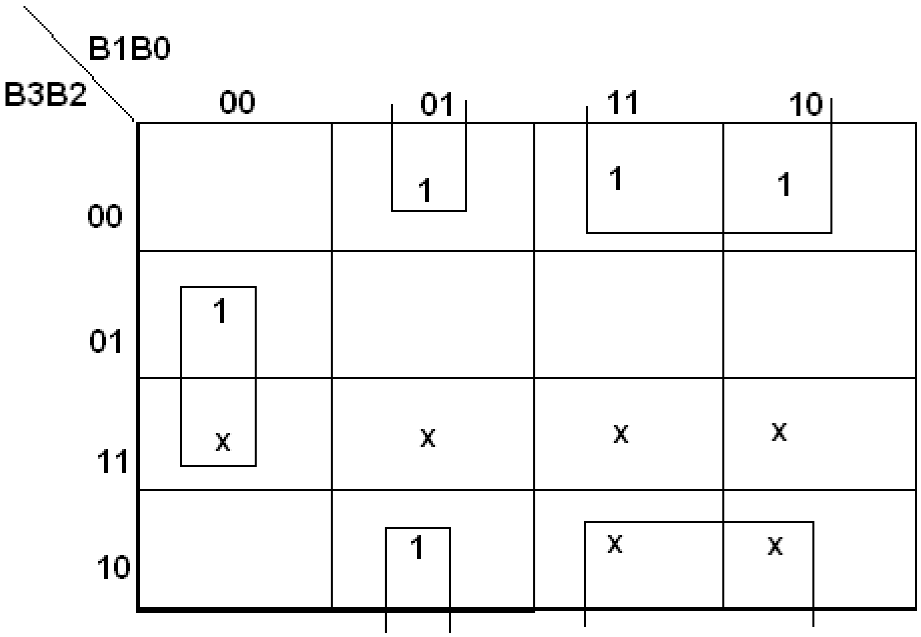
K-Map for E_3:



B3B2 \ B1B0	00	01	11	10
00				
01		1	1	1
11	x	x	x	x
10	1	1	x	x

E3 = B3 + B2 (B0 + B1)

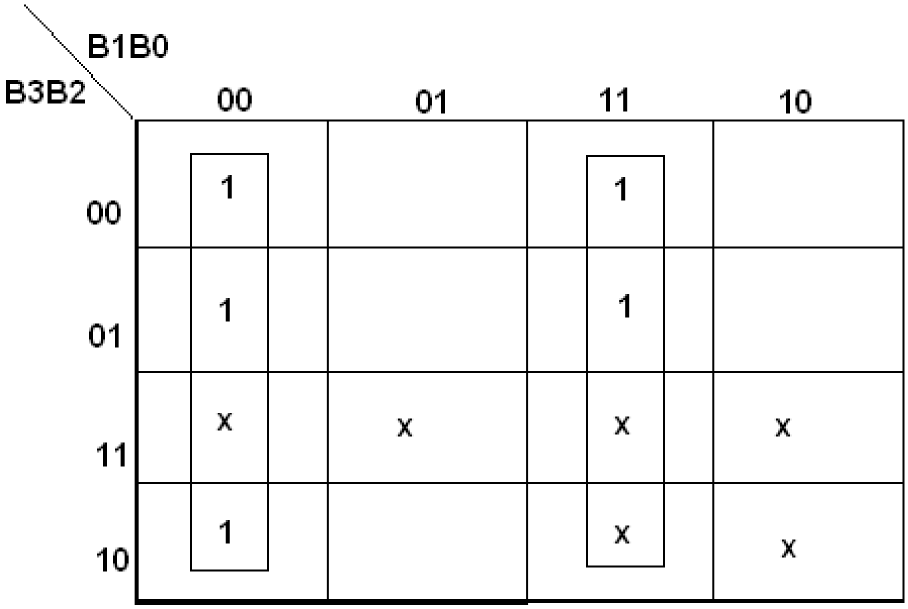
K-Map for E_2:



B3B2 \ B1B0	00	01	11	10
00		1	1	1
01	1			
11	x	x	x	x
10		1	x	x

$$E2 = B2 \oplus (B1 + B0)$$

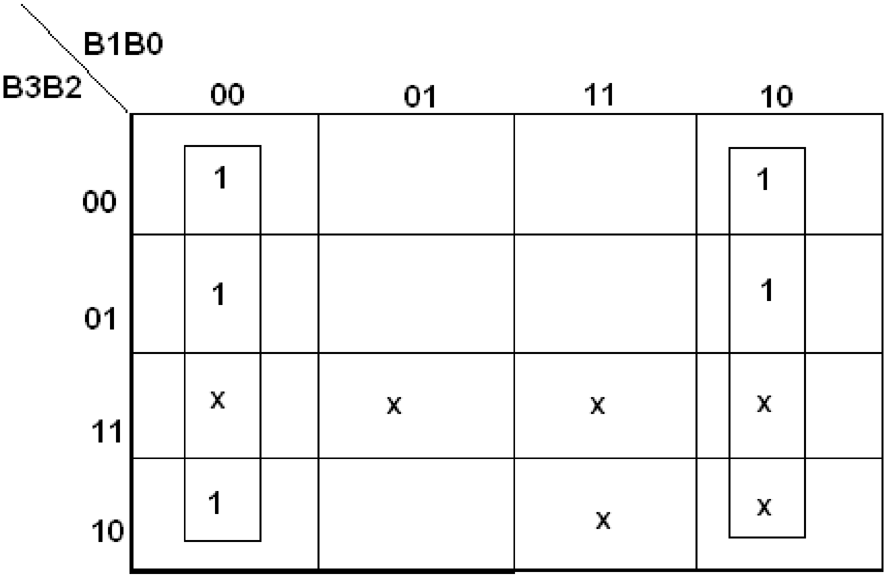
K-Map for E_1:



B3B2 \ B1B0	00	01	11	10
00	1		1	
01	1		1	
11	x	x	x	x
10	1		x	x

$$E1 = B1 \overline{\oplus} B0$$

K-Map for E_0:



B3B2 \ B1B0	00	01	11	10
00	1			1
01	1			1
11	x	x	x	x
10	1		x	x

$$E0 = \overline{B0}$$

TRUTH TABLE:

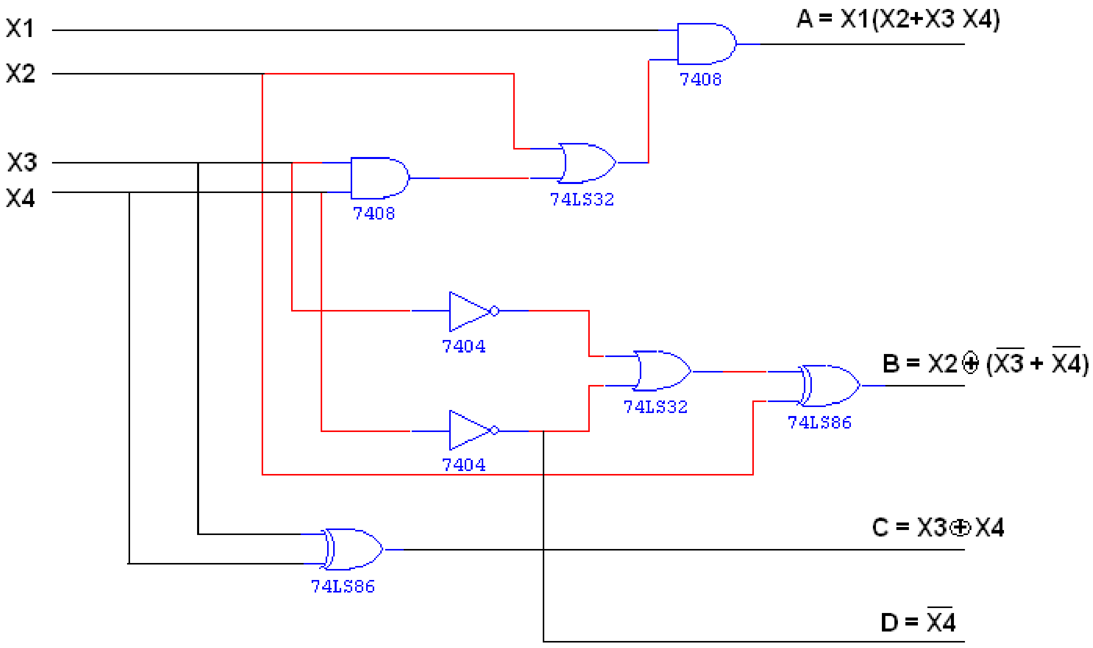
| BCD input | Excess – 3 output
|

B3	*B2*	*B1*	*B0*	*G 3*	*G 2*	*G 1*	*G 0*
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*

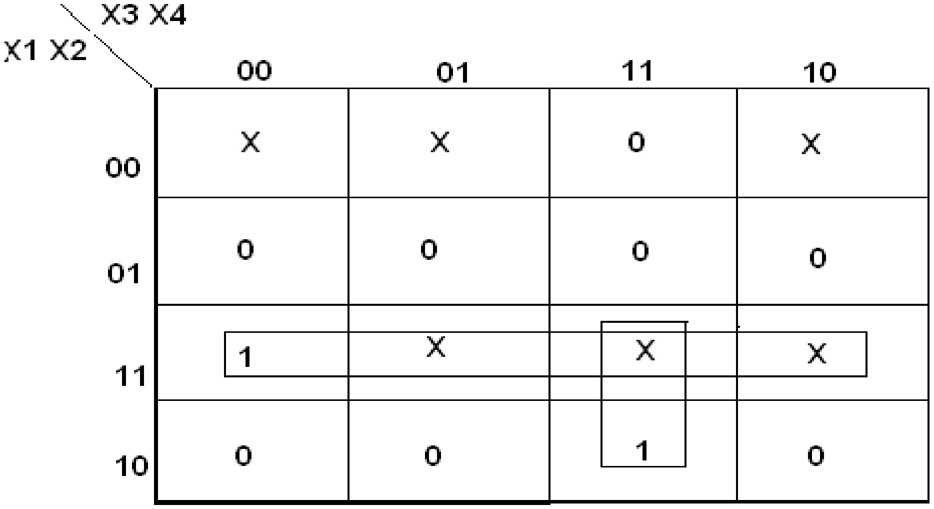
1	*1*	*1*	*0*	*x*	*x*	*x*	*x*
1	*1*	*1*	*1*	*x*	*x*	*x*	*x*

LOGIC DIAGRAM:

EXCESS-3 TO BCD CONVERTOR



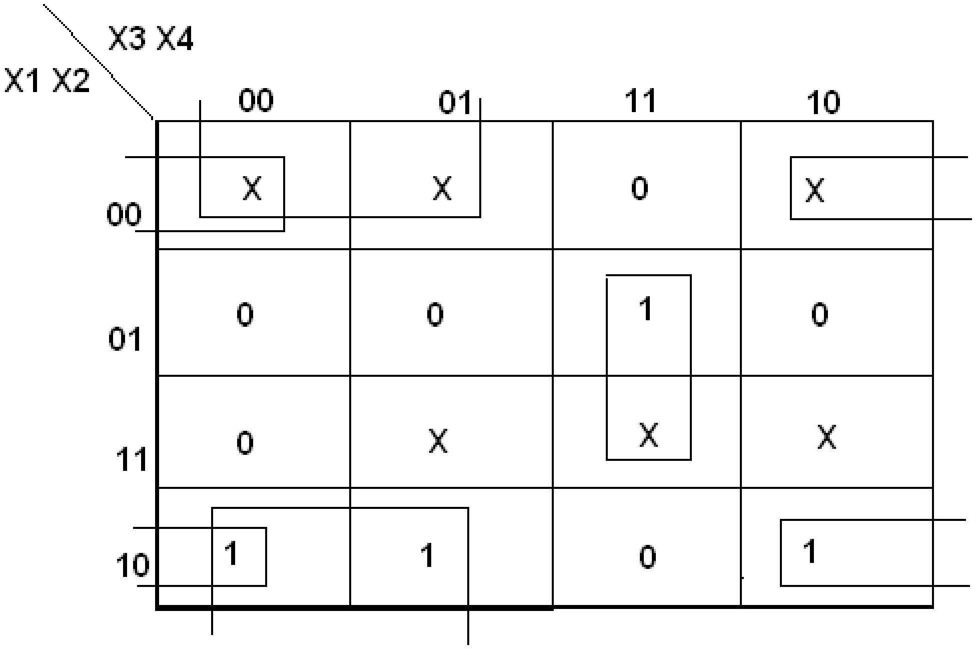
K-Map for A:



X1 X2 \ X3 X4	00	01	11	10
00	X	X	0	X
01	0	0	0	0
11	1	X	X	X
10	0	0	1	0

A = X1 X2 + X3 X4 X1

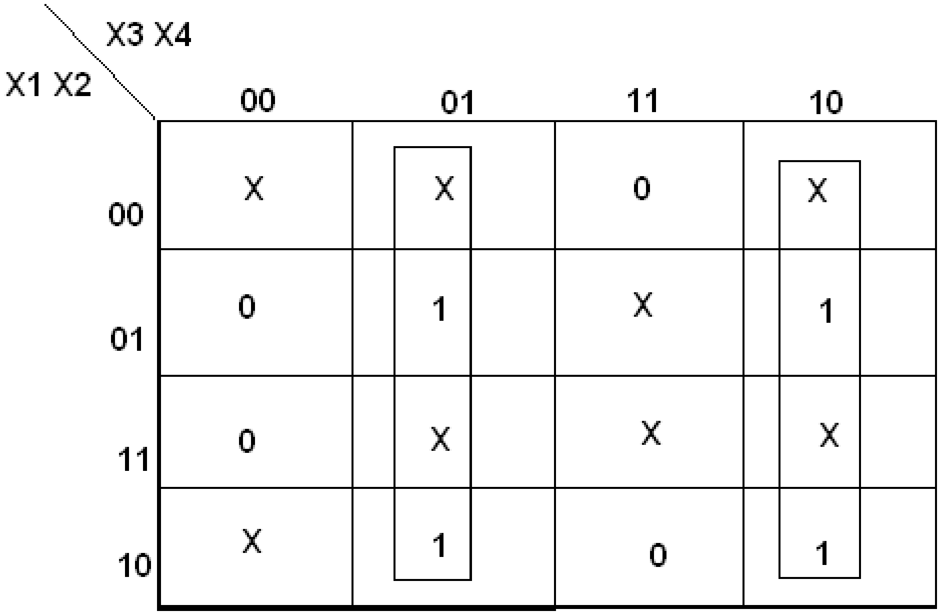
K-Map for B:



X1 X2 \ X3 X4	00	01	11	10
00	X	X	0	X
01	0	0	1	0
11	0	X	X	X
10	1	1	0	1

$$B = X2 \oplus (\overline{X3} + \overline{X4})$$

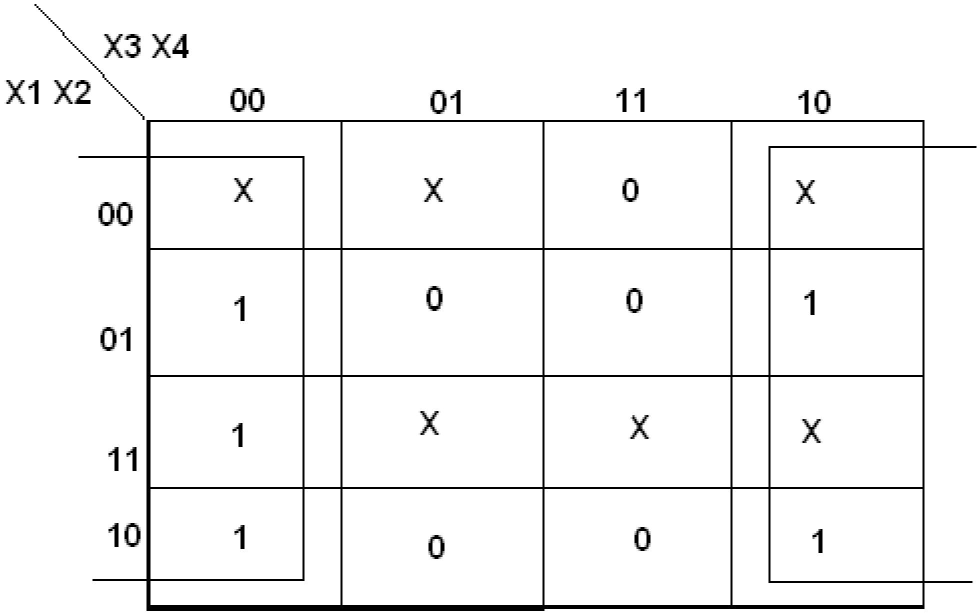
K-Map for C:



X1 X2 \ X3 X4	00	01	11	10
00	X	X	0	X
01	0	1	X	1
11	0	X	X	X
10	X	1	0	1

$C = X3 \oplus X4$

K-Map for D:



X1 X2 \ X3 X4	00	01	11	10
00	X	X	0	X
01	1	0	0	1
11	1	X	X	X
10	1	0	0	1

$D = \overline{X4}$

TRUTH TABLE:

| Excess – 3 Input | BCD Output
|

B3	B2	B1	B0	G3	G2	G1	G0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1

PROCEDURE:

(i) Connections were given as per circuit diagram.

(ii) Logical inputs were given as per truth table

(iii) Observe the logical output and verify with the truth tables.

RESULT:

EXPT NO. : DESIGN OF 4-BIT ADDER AND SUBTRACTORDATE:

AIM:

To design and implement 4-bit adder and subtractor using IC 7483.

APPARATUS REQUIRED:

Sl.No.	*COMPONENT*	*SPECIFICATION*	*QTY.*
1.	*IC*	*IC 7483*	*1*
2.	*EX-OR GATE*	*IC 7486*	*1*
3.	*NOT GATE*	*IC 7404*	*1*
3.	*IC TRAINER KIT*	*-*	*1*
4.	*PATCH CORDS*	*-*	*40*

THEORY:

4 BIT BINARY ADDER:

A binary adder is a digital circuit that produces the arithmetic sum of two binary numbers. It can be constructed with full adders connected in cascade, with the output carry from each full adder connected to the input carry of next full adder in chain. The augends bits of 'A' and the addend bits of 'B' are designated by subscript numbers from right to left, with subscript 0 denoting the least significant bits. The carries are connected in chain through the full adder. The input carry to the adder is C_0 and it ripples through the full adder to the output carry C_4.

4 BIT BINARY SUBTRACTOR:

The circuit for subtracting A-B consists of an adder with inverters, placed between each data input 'B' and the corresponding input of full adder. The input carry C_0 must be equal to 1 when performing subtraction.

4 BIT BINARY ADDER/SUBTRACTOR:

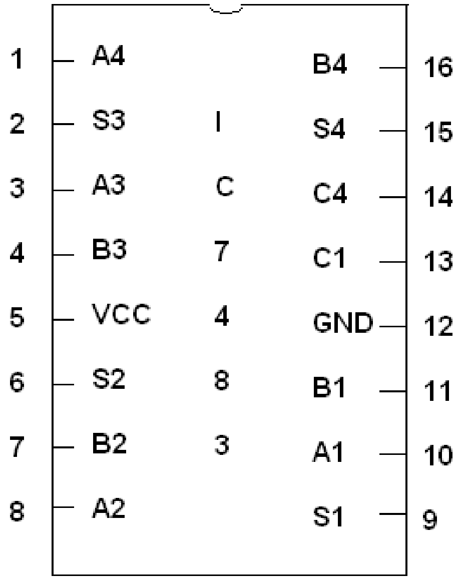
The addition and subtraction operation can be combined into one circuit with one common binary adder. The mode input M controls the operation. When M=0, the circuit is adder circuit. When M=1, it becomes subtractor.

4 BIT BCD ADDER:

Consider the arithmetic addition of two decimal digits in BCD, together with an input carry from a previous stage. Since each input digit does not exceed 9, the output sum cannot be greater than 19, the 1 in the sum being an input carry. The output of two decimal digits must be represented in BCD and should appear in the form *listed in the columns.*

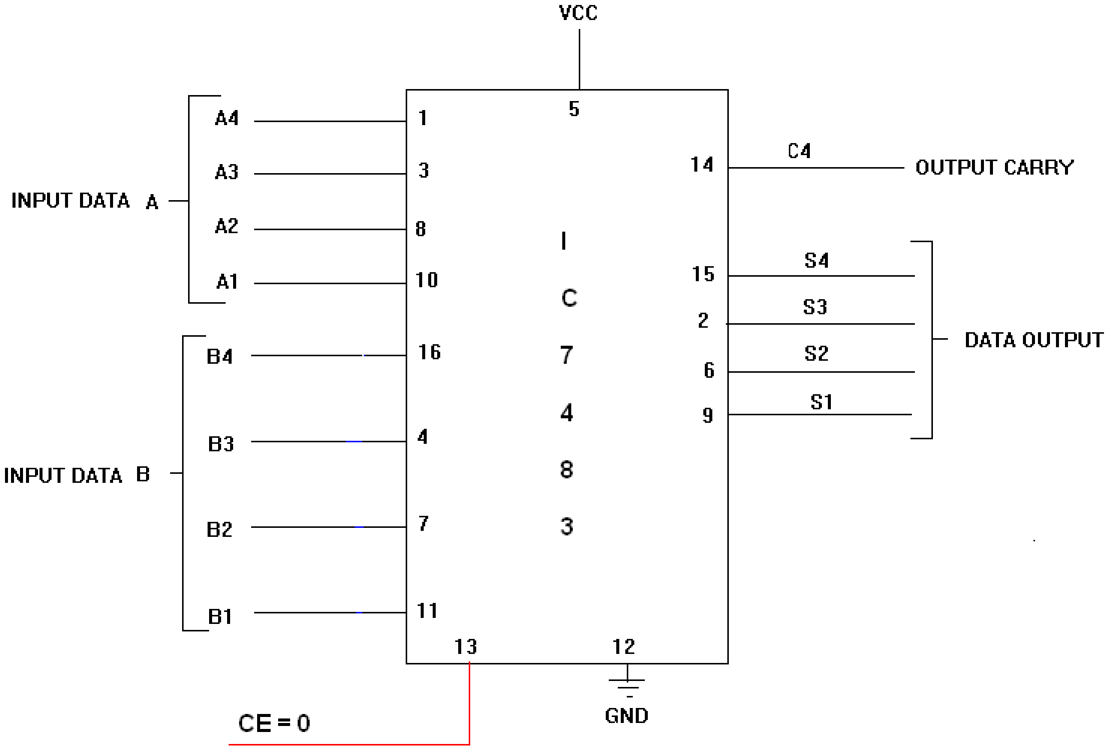
ABCD adder that adds 2 BCD digits and produce a sum digit in BCD. The 2 decimal digits, together with the input carry, are first added in the top 4 bit adder to produce the binary sum.

PIN DIAGRAM FOR IC 7483:



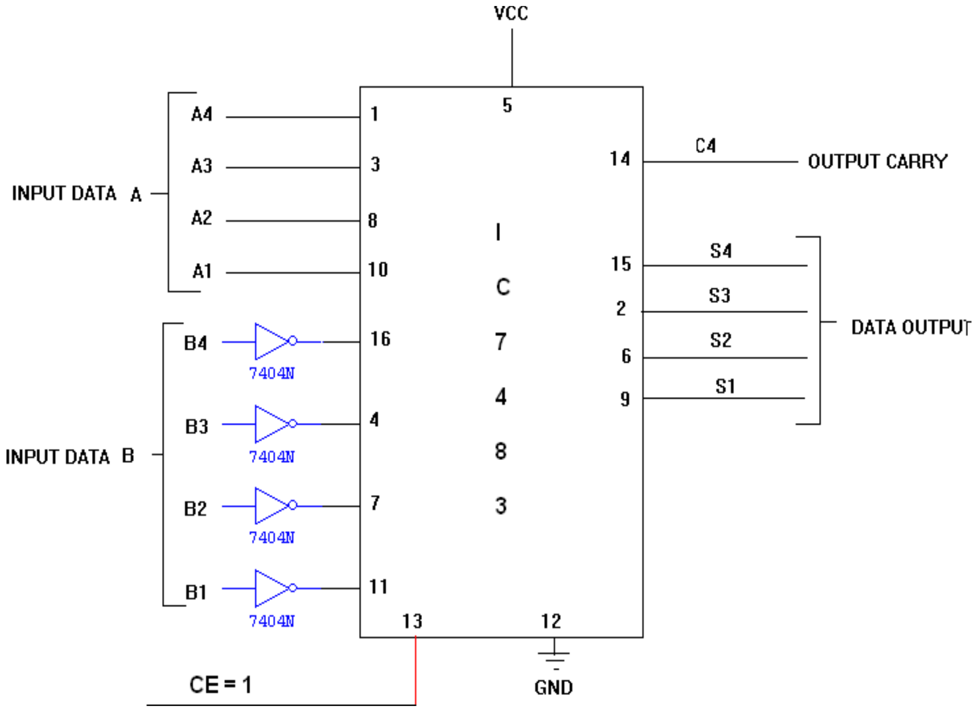
LOGIC DIAGRAM:

4- BIT BINARY ADDER



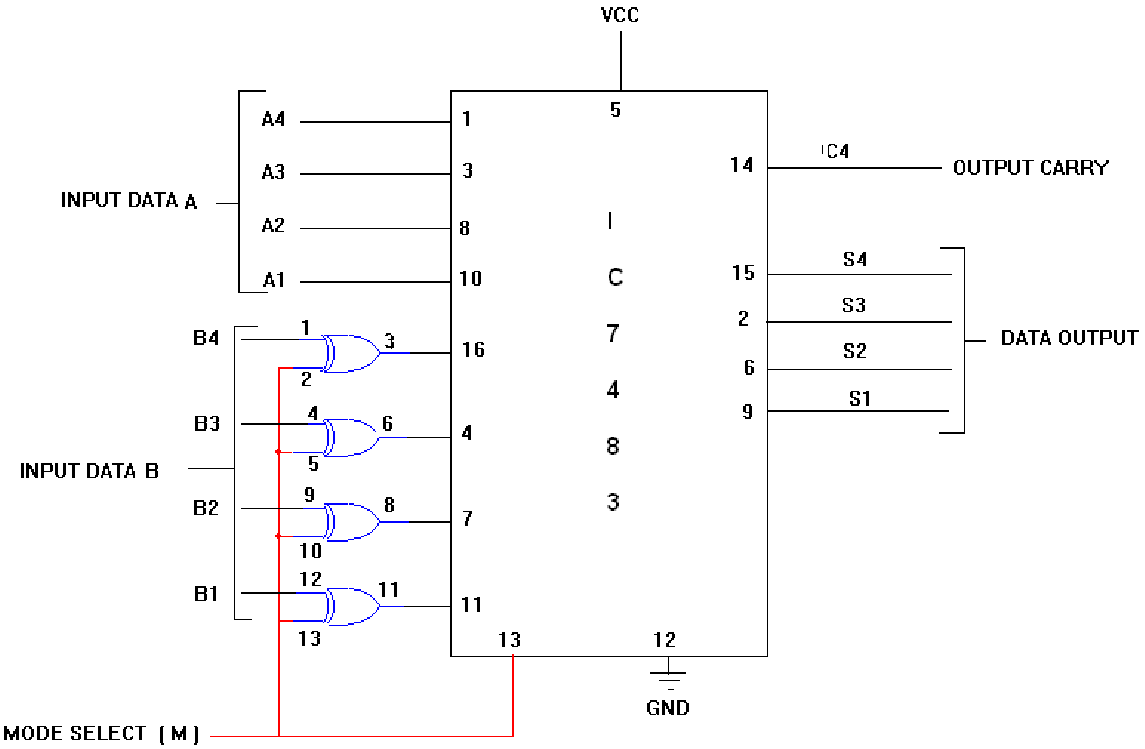
LOGIC DIAGRAM:

4-BIT BINARY SUBTRACTOR



LOGIC DIAGRAM:

4- BIT BINARY ADDER/SUBTRACTOR



M=0 (ADDITION)

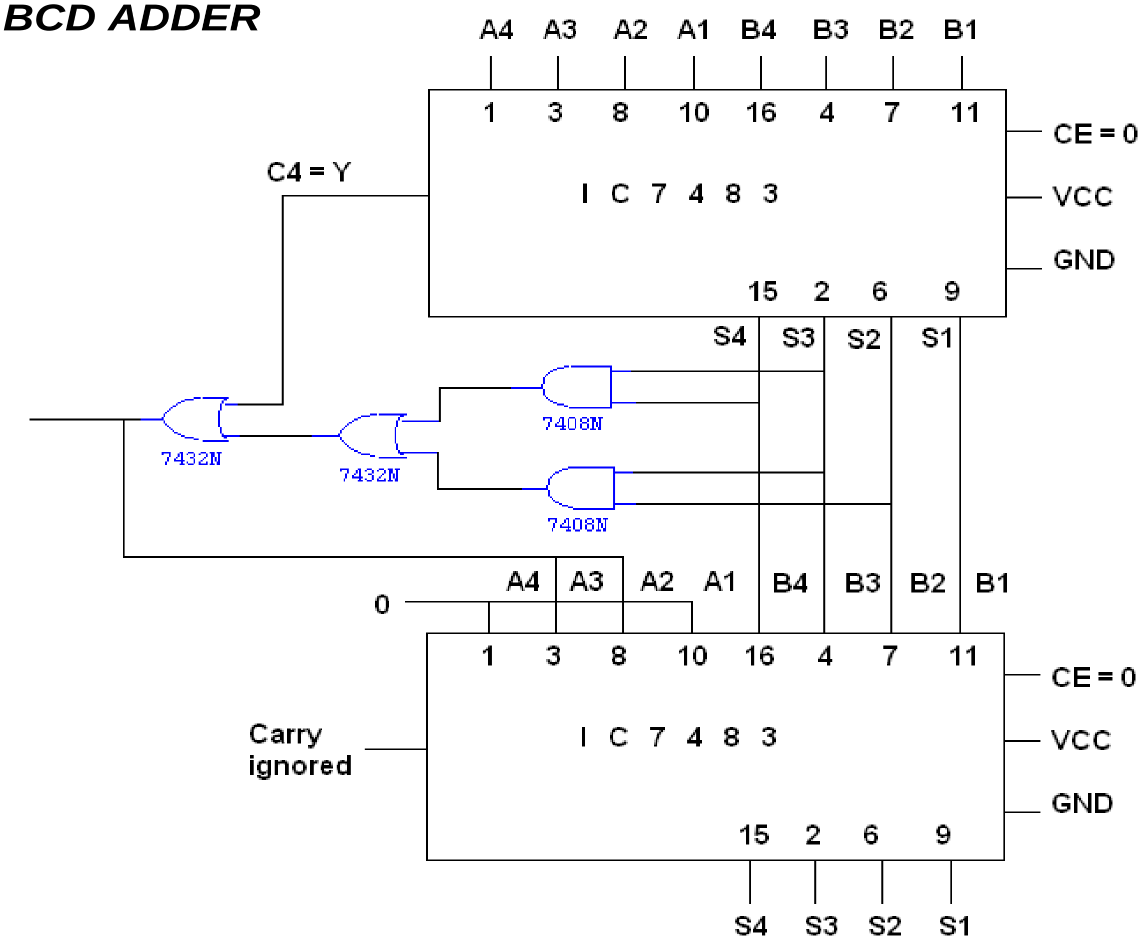
M=1 (SUBTRACTION)

TRUTH TABLE:

Input Data A				Input Data B				Addition					Subtraction				
A4	A3	A2	A1	B4	B3	B2	B1	C	S4	S3	S2	S1	B	D4	D3	D2	D1
1	0	0	0	0	0	1	0	0	1	0	1	0	1	0	1	1	0
1	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	1	0	0	0	0	1	0	1	0	0	1	0	1	0
0	0	0	1	0	1	1	1	0	1	0	0	0	0	1	0	1	0
1	0	1	0	1	0	1	1	1	0	0	1	0	0	1	1	1	1
1	1	1	0	1	1	1	1	1	1	0	1	0	0	1	1	1	1
1	0	1	0	1	1	0	1	1	0	1	1	1	0	1	1	0	1

LOGIC DIAGRAM:

BCD ADDER



K MAP

S3 S4 \ S1 S2	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

$$Y = S4\,(S3 + S2)$$

TRUTH TABLE:

BCD SUM				CARRY
S4	S3	S2	S1	C
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

PROCEDURE:

(i) Connections were given as per circuit diagram.

(ii) Logical inputs were given as per truth table

(iii) Observe the logical output and verify with the truth tables.

RESULT:

EXPT NO.:

DATE:

DESIGN AND IMPLEMENTATION OF MAGNITUDE COMPARATOR

AIM

: *To design and implement*

(i) 2 – bit magnitude comparator using basic gates.

(ii) 8 – bit magnitude comparator using IC 7485.

APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	AND GATE	IC 7408	2
2.	X-OR GATE	IC 7486	1
3.	OR GATE	IC 7432	1
4.	NOT GATE	IC 7404	1
5.	4-BIT MAGNITUDE COMPARATOR	IC 7485	2
6.	IC TRAINER KIT	-	1
7.	PATCH CORDS	-	30

THEORY:

The comparison of two numbers is an operator that determine one number is greater than, less than (or) equal to the other number. A magnitude comparator is a combinational circuit that compares two numbers A and B and determine their relative magnitude. The outcome of the comparator is specified by three binary variables that indicate whether A>B, A=B (or) A<B.

$A = A_3\ A_2\ A_1\ A_0B$

$= B_3\ B_2\ B_1\ B_0$

The equality of the two numbers and B is displayed in a combinational circuit designated by the symbol (A=B).

This indicates A greater than B, then inspect the relative magnitude of pairs of significant digits starting from most significant position. A is 0 and that of B is 0.

We have A<B, the sequential comparison can be expanded as

$A>B = A3B_3^{1} + X_3A_2B_2^{1} + X_3X_2A_1B_1^{1} + X_3X_2X_1A_0B_0^{1}$

$A<B = A_3^{1}B_3 + X_3A_2^{1}B_2 + X_3X2A_1^{1}B_1 + X_3X_2X_1A_0^{1}B_0$

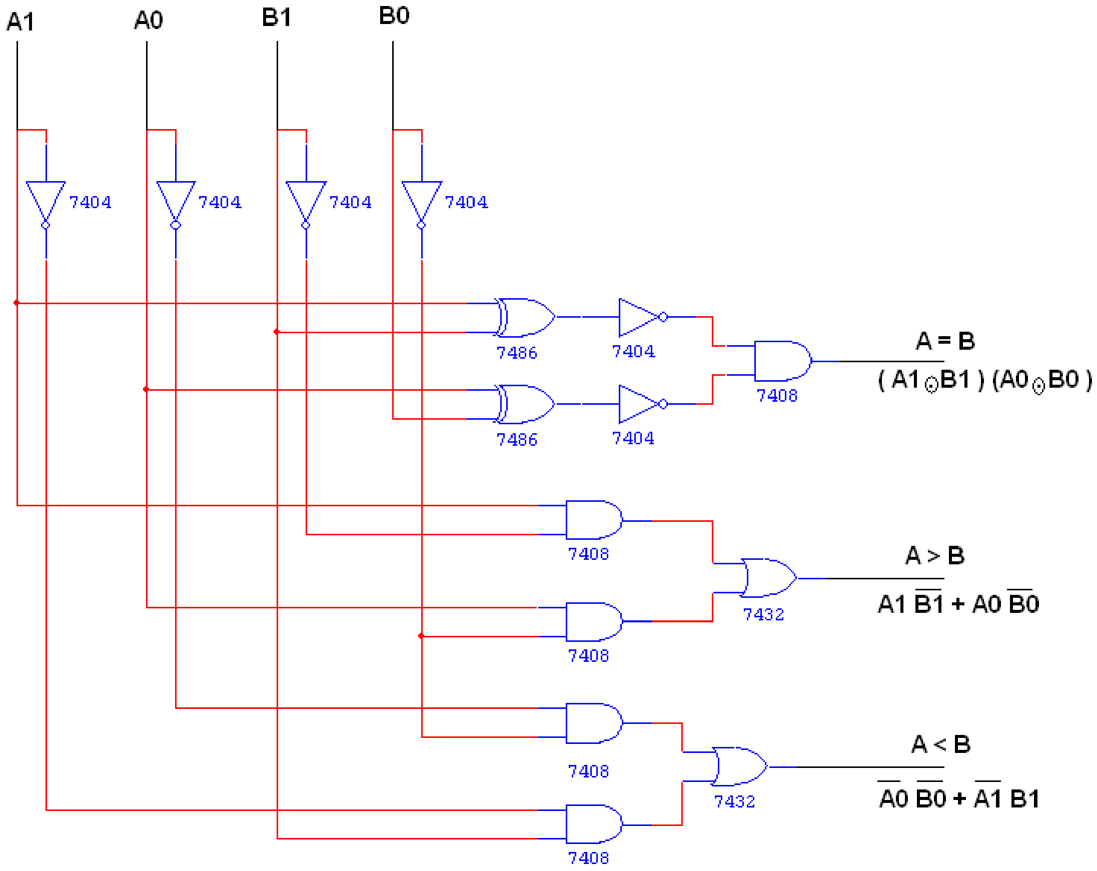
The same circuit can be used to compare the relative magnitude of two BCD digits.

Where, A = B is expanded as,

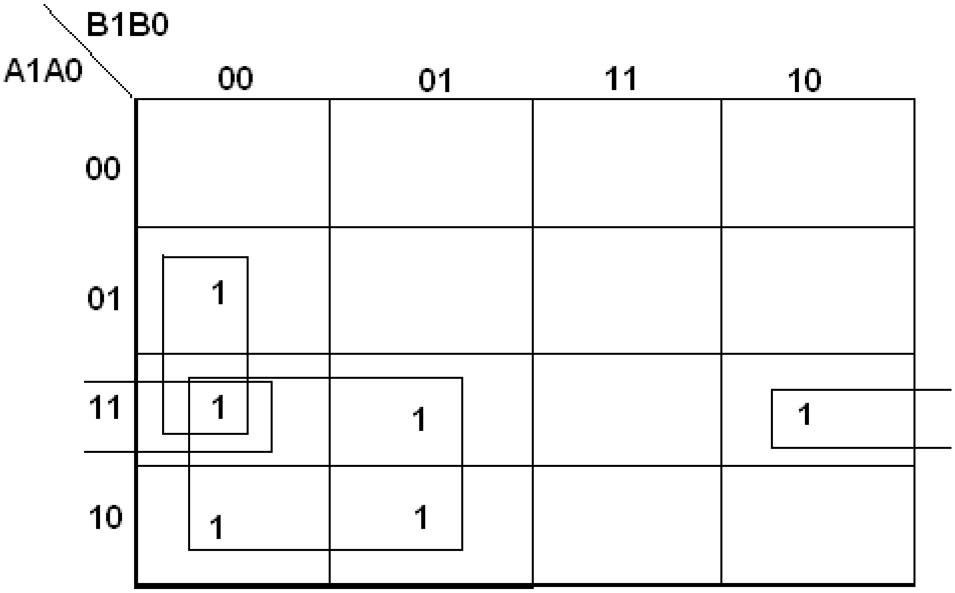
$A = B = (A_3 + B_3)\ (A_2 + B_2)\ (A_1 + B_1)\quad (A_0 + B_0)$

$x_3 \qquad x_2 \qquad x_1 \qquad x_0$

LOGIC DIAGRAM:
2 BIT MAGNITUDE COMPARATOR

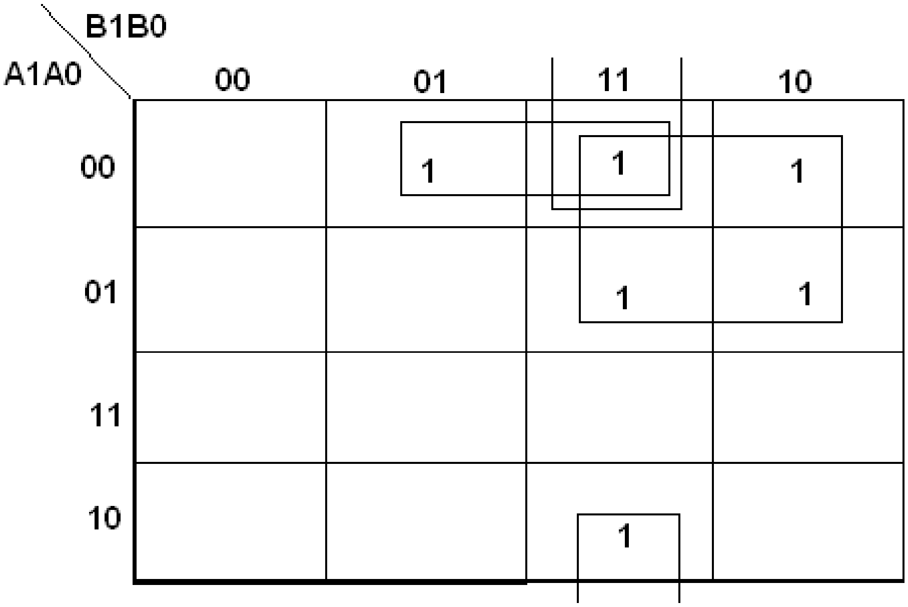


K MAP



B1B0 A1A0	00	01	11	10
00				
01	1			
11	1	1		1
10	1	1		

$A > B = A0\ \overline{B0}\ B1 + A1\ \overline{B1} + A1\ A0\ \overline{B0}$



B1B0 A1A0	00	01	11	10
00		1	1	1
01			1	1
11				
10			1	

$A < B = \overline{A1}\ \overline{A0}\ B0 + \overline{A0}\ B0\ B1 + \overline{A1}\ B1$

A1A0 \ B1B0	00	01	11	10
00	1			
01		1		
11			1	
10				1

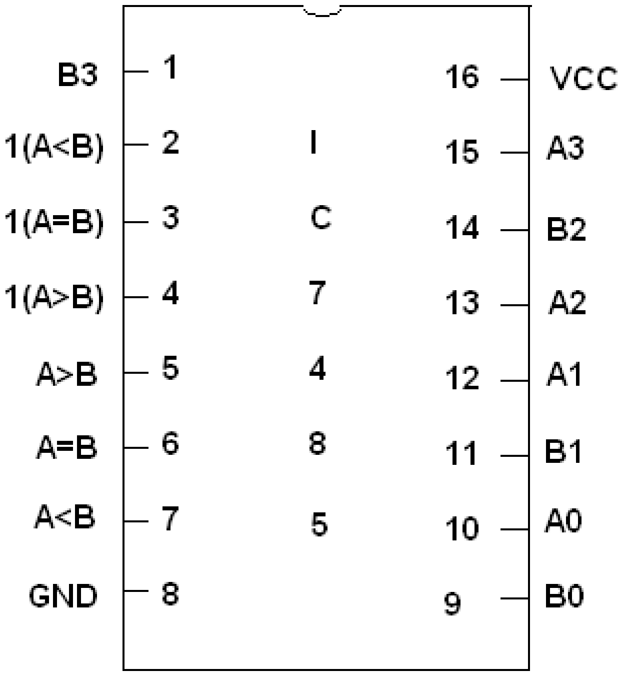
$A = B = (A0 \odot B0)(A1 \odot B1)$

TRUTH TABLE

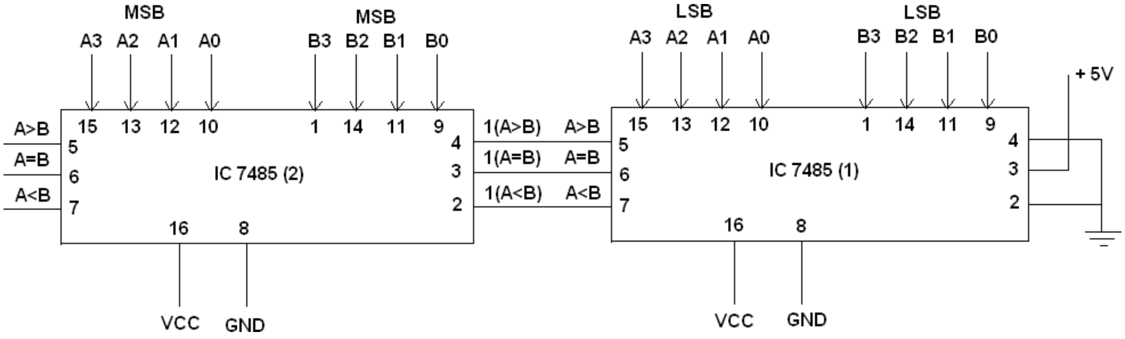
A 1	*A 0*	*B 1*	*B 0*	*A > B*	*A = B*	*A < B*
0	*0*	*0*	*0*	*0*	*1*	*0*
0	*0*	*0*	*1*	*0*	*0*	*1*
0	*0*	*1*	*0*	*0*	*0*	*1*
0	*0*	*1*	*1*	*0*	*0*	*1*
0	*1*	*0*	*0*	*1*	*0*	*0*
0	*1*	*0*	*1*	*0*	*1*	*0*
0	*1*	*1*	*0*	*0*	*0*	*1*
0	*1*	*1*	*1*	*0*	*0*	*1*
1	*0*	*0*	*0*	*1*	*0*	*0*
1	*0*	*0*	*1*	*1*	*0*	*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***

PIN DIAGRAM FOR IC 7485:



LOGIC DIAGRAM:
8 BIT MAGNITUDE COMPARATOR



TRUTH TABLE:

A	B	A>B	A=B	A<B
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0	1	0
0 0 0 1 0 0 0 1	0 0 0 0 0 0 0 0	1	0	0
0 0 0 0 0 0 0 0	0 0 0 1 0 0 0 1	0	0	1

PROCEDURE:

(i) Connections are given as per circuit diagram.

(ii) Logical inputs are given as per circuit diagram.

(iii) Observe the output and verify the truth table.

RESULT:

EXPT NO.

:DATE:

16 BIT ODD/EVEN PARITY CHECKER /GENERATOR

AIM

: *To design and implement 16 bit odd/even parity checker generator using IC 74180.*

APPARATUS REQUIRED:

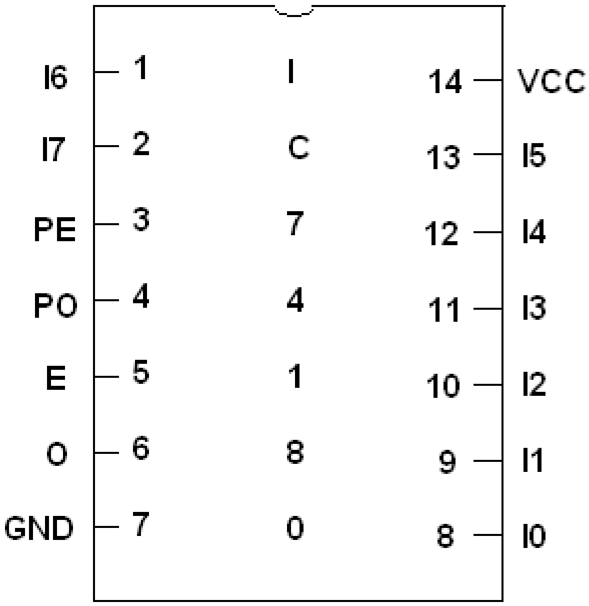
Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	NOT GATE	IC 7404	1
1.		IC 74180	2
2.	IC TRAINER KIT	-	1
3.	PATCH CORDS	-	30

THEORY:

A parity bit is used for detecting errors during transmission of binary information. A parity bit is an extra bit included with a binary message to make the number is either even or odd. The message including the parity bit is transmitted and then checked at the receiver ends for errors. An error is detected if the checked parity bit doesn't correspond to the one transmitted. The circuit that generates the parity bit in the transmitter is called a 'parity generator' and the circuit that checks the parity in the receiver is called a 'parity checker'.

In even parity, the added parity bit will make the total number is even amount. In odd parity, the added parity bit will make the total number is odd amount. The parity checker circuit checks for possible errors in the transmission. If the information is passed in even parity, then the bits required must have an even number of 1's. An error occur during transmission, if the received bits have an odd number of 1's indicating that one bit has changed in value during transmission.

PIN DIAGRAM FOR IC 74180:



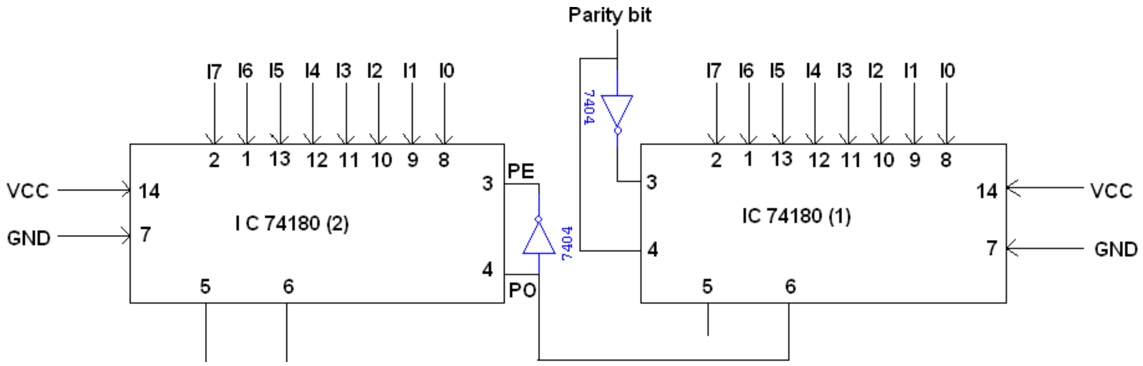
FUNCTION TABLE:

INPUTS			***OUTPUTS***	
Number of High Data Inputs (I0 – I7)	***PE***	***PO***	***∑E***	***∑O***
EVEN	***1***	***0***	***1***	***0***
ODD	***1***	***0***	***0***	***1***
EVEN	***0***	***1***	***0***	***1***

ODD	***0***	***1***	***1***	***0***
X	***1***	***1***	***0***	***0***
X	***0***	***0***	***1***	***1***

LOGIC DIAGRAM:

16 BIT ODD/EVEN PARITY CHECKER

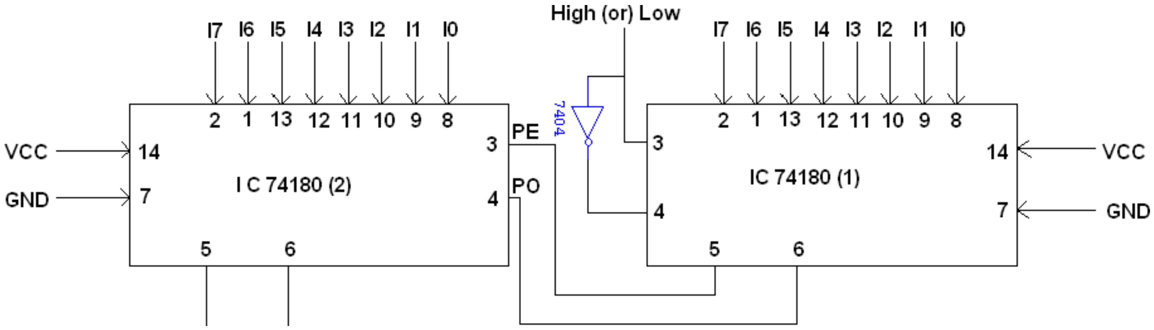


TRUTH TABLE:

I7 I6 I5 I4 I3 I2 I1 I0	***I7'I6'I5'I4'I3'I2'11' I0'***	***Active***	***∑E***	***∑O***
0 0 0 0 0 0 0 1	***0 0 0 0 0 0 0 0***	***1***	***1***	***0***
0 0 0 0 0 1 1 0	***0 0 0 0 0 1 1 0***	***0***	***1***	***0***
0 0 0 0 0 1 1 0	***0 0 0 0 0 1 1 0***	***1***	***0***	***1***

LOGIC DIAGRAM:

16 BIT ODD/EVEN PARITY GENERATOR



TRUTH TABLE:

I7 I6 I5 I4 I3 I2 I1 I0	*I7 I6 I5 I4 I3 I2 I1 I0*	*Active*	*∑E*	*∑O*
1 1 0 0 0 0 0 0	*1 1 0 0 0 0 0 0*	*1*	*1*	*0*
1 1 0 0 0 0 0 0	*1 1 0 0 0 0 0 0*	*0*	*0*	*1*
1 1 0 0 0 0 0 0	*0 1 0 0 0 0 0 0*	*0*	*1*	*0*

PROCEDURE:

(i) Connections are given as per circuit diagram.

(ii) Logical inputs are given as per circuit diagram.

(iii) Observe the output and verify the truth table.

RESULT:

EXPT NO.

:DATE:

DESIGN AND IMPLEMENTATION OF MULTIPLEXER AND DEMULTIPLEXER

AIM

: *To design and implement multiplexer and demultiplexer using logic gates and study of IC 74150 and IC 74154.*

APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	3 I/P AND GATE	IC 7411	2
2.	OR GATE	IC 7432	1
3.	NOT GATE	IC 7404	1
2.	IC TRAINER KIT	-	1
3.	PATCH CORDS	-	32

THEORY:

MULTIPLEXER:

Multiplexer means transmitting a large number of information units over a smaller number of channels or lines. A digital multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line. The selection of a particular input line is controlled by a set of selection lines.

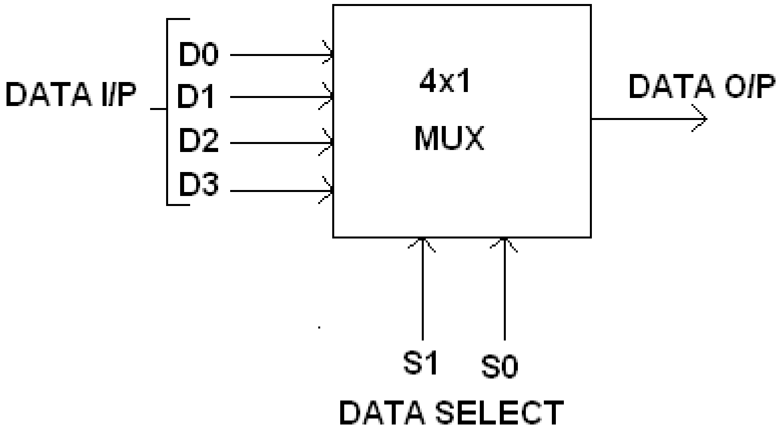
Normally there are 2^n input line and n selection lines whose bit combination determine which input is selected.

DEMULTIPLEXER:

The function of Demultiplexer is in contrast to multiplexer function. It takes information from one line and distributes it to a given number of output lines. For this reason, the demultiplexer is also known as a data distributor. Decoder can also be used as demultiplexer.

In the 1: 4 demultiplexer circuit, the data input line goes to all of the AND gates. The data select lines enable only one gate at a time and the data on the data input line will pass through the selected gate to the associated data output line.

BLOCK DIAGRAM FOR 4:1 MULTIPLEXER:

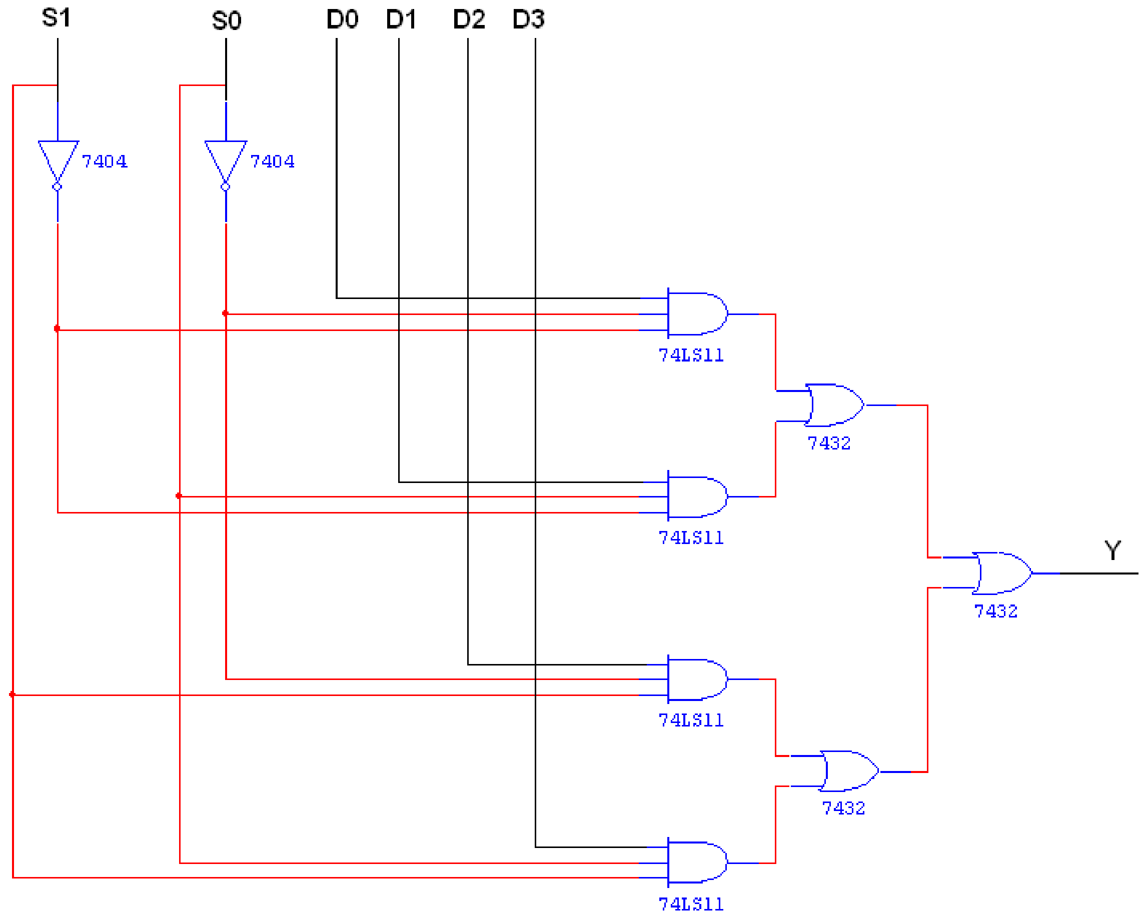


FUNCTION TABLE:

S1	*S0*	*INPUTS Y*

0	***0***	***D0 → D0 S1’ S0’***
0	***1***	***D1 → D1 S1’ S0***
1	***0***	***D2 → D2 S1 S0’***
1	***1***	***D3 → D3 S1 S0***

Y = D0 S1’ S0’ + D1 S1’ S0 + D2 S1 S0’ + D3 S1 S0CIRCUIT DIAGRAM FOR MULTIPLEXER:

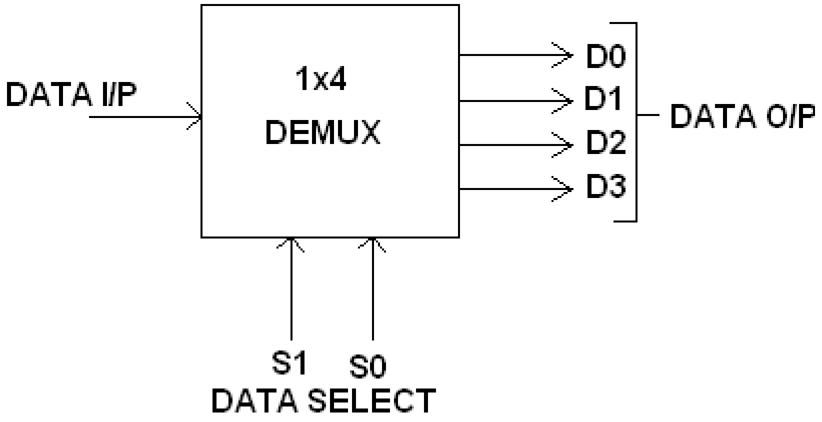


TRUTH TABLE:

S1	***S0***	***Y = OUTPUT***

0	0	D0
0	1	D1
1	0	D2
1	1	D3

BLOCK DIAGRAM FOR 1:4 DEMULTIPLEXER:

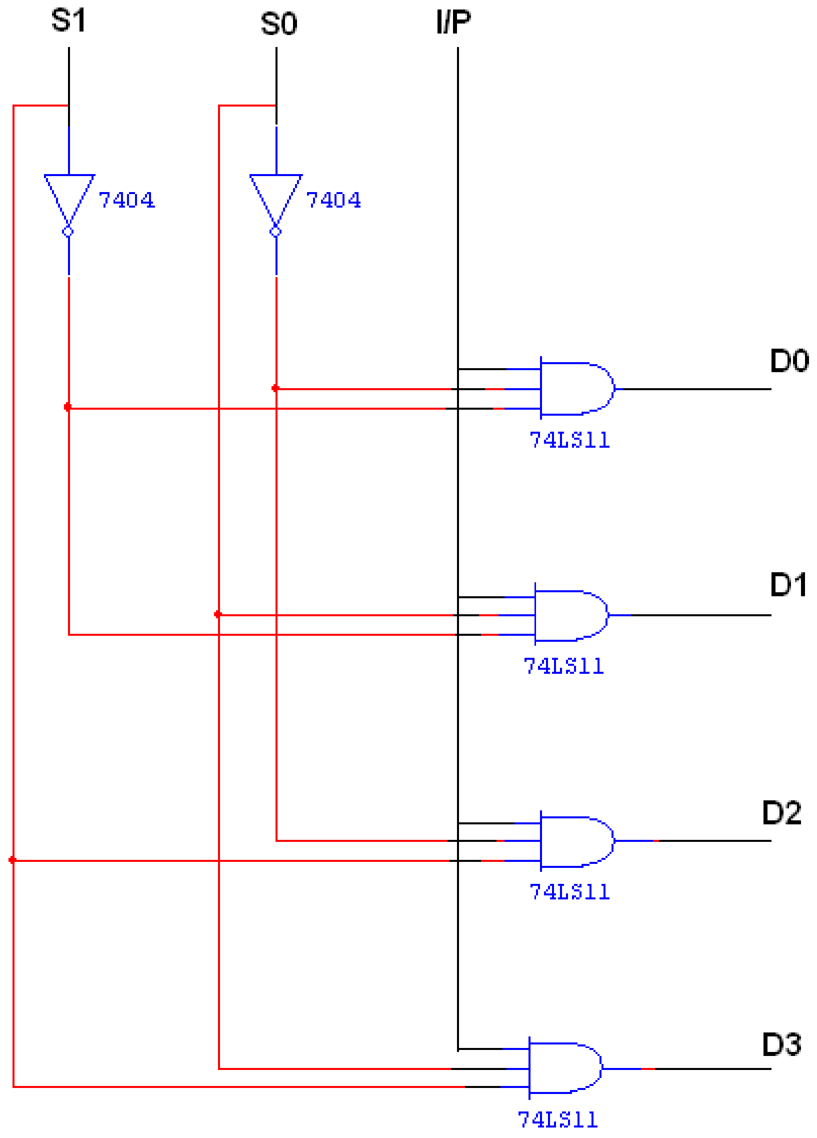


FUNCTION TABLE:

S1	S0	INPUT
0	0	X → D0 = X S1’ S0’
0	1	X → D1 = X S1’ S0
1	0	X → D2 = X S1 S0’
1	1	X → D3 = X S1 S0

Y = X S1’ S0’ + X S1’ S0 + X S1 S0’ + X S1 S0

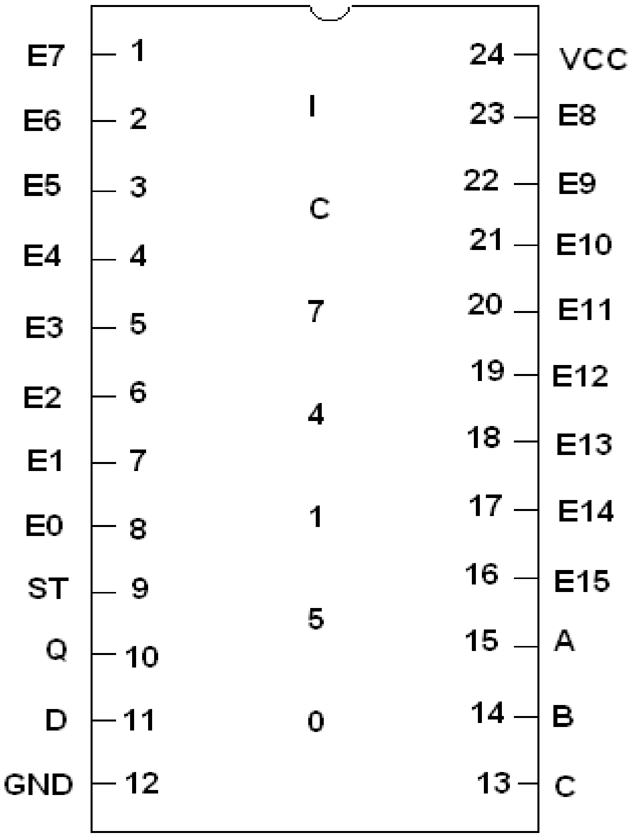
LOGIC DIAGRAM FOR DEMULTIPLEXER:



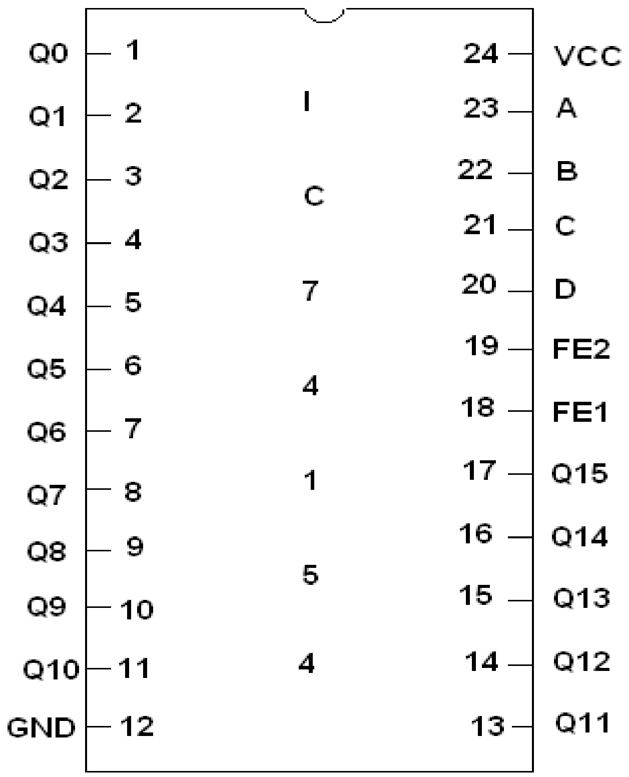
TRUTH TABLE:

INPUT			*OUTPUT*			
S1	*S0*	*I/P*	*D0*	*D1*	*D2*	*D3*
0	*0*	*0*	*0*	*0*	*0*	*0*
0	*0*	*1*	*1*	*0*	*0*	*0*
0	*1*	*0*	*0*	*0*	*0*	*0*
0	*1*	*1*	*0*	*1*	*0*	*0*
1	*0*	*0*	*0*	*0*	*0*	*0*
1	*0*	*1*	*0*	*0*	*1*	*0*
1	*1*	*0*	*0*	*0*	*0*	*0*
1	*1*	*1*	*0*	*0*	*0*	*1*

PIN DIAGRAM FOR IC 74150:



PIN DIAGRAM FOR IC 74154:



PROCEDURE:

(i) Connections are given as per circuit diagram.

(ii) Logical inputs are given as per circuit diagram.

(iii) Observe the output and verify the truth table.

RESULT:

EXPT NO.:

DATE:

DESIGN AND IMPLEMENTATION OF ENCODER AND DECODER

AIM:

To design and implement encoder and decoder using logic gates and study of IC 7445 and IC 74147.

APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	3 I/P NAND GATE	IC 7410	2
2.	OR GATE	IC 7432	3
3.	NOT GATE	IC 7404	1
2.	IC TRAINER KIT	-	1
3.	PATCH CORDS	-	27

THEORY:

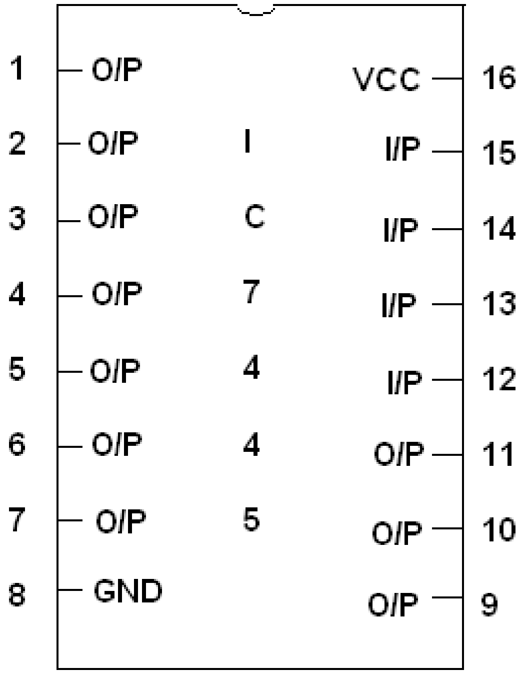
ENCODER:

An encoder is a digital circuit that perform inverse operation of a decoder. An encoder has 2^n input lines and n output lines. In encoder the output lines generates the binary code corresponding to the input value. In octal to binary encoder it has eight inputs, one for each octal digit and three output that generate the corresponding binary code. In encoder it is assumed that only one input has a value of one at any given time otherwise the circuit is meaningless. It has an ambiguila that when all inputs are zero the outputs are zero. The *zero outputs can also be generated when D0 = 1.*

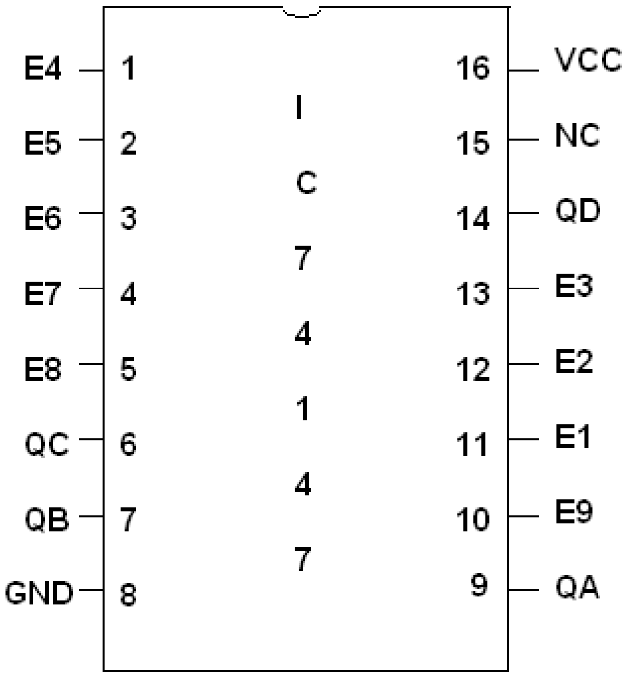
DECODER:

A decoder is a multiple input multiple output logic circuit which converts coded input into coded output where input and output codes are different. The input code generally has fewer bits than the output code. Each input code word produces a different output code word i.e there is one to one mapping can be expressed in truth table. In the block diagram of decoder circuit the encoded information is present as n input producing 2^n possible outputs. 2^n output values are from 0 through out 2^n – 1.

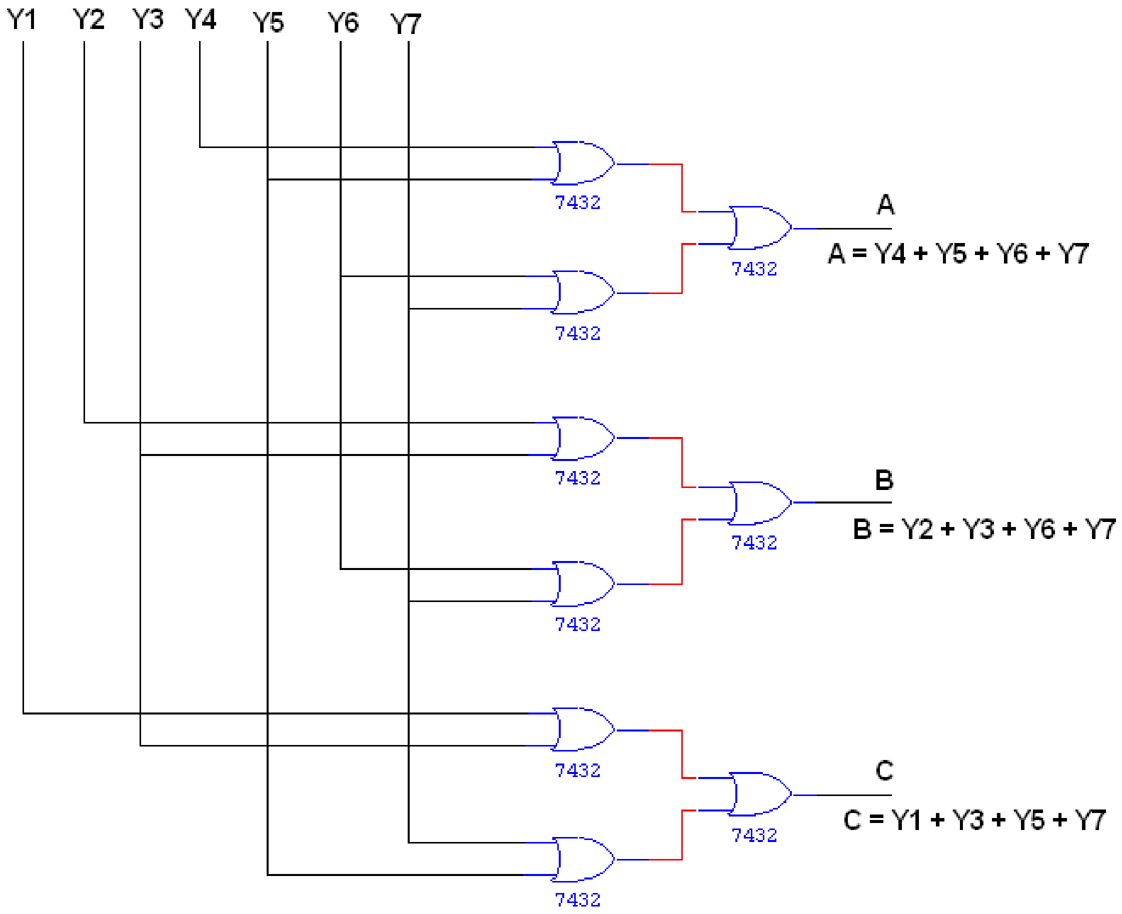
PIN DIAGRAM FOR IC 7445: BCD TO DECIMAL DECODER:



PIN DIAGRAM FOR IC 74147:



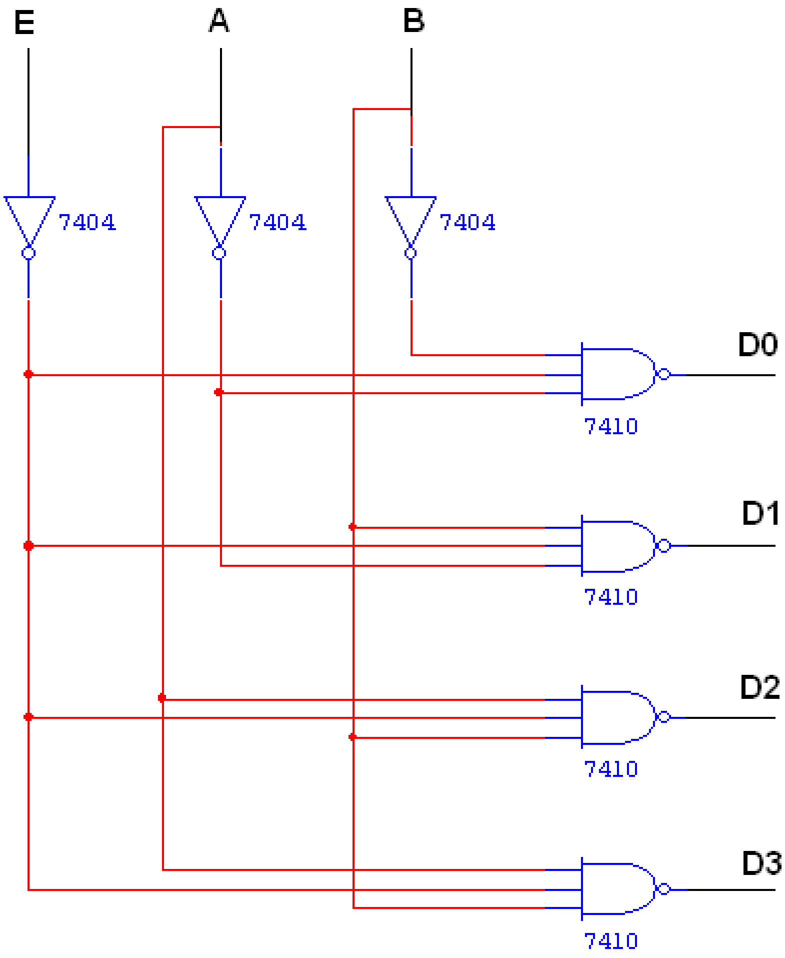
LOGIC DIAGRAM FOR ENCODER:



TRUTH TABLE:

INPUT							***OUTPUT***		
Y1	***Y2***	***Y3***	***Y4***	***Y5***	***Y6***	***Y7***	***A***	***B***	***C***
1	***0***	***0***	***0***	***0***	***0***	***0***	***0***	***0***	***1***
0	***1***	***0***	***0***	***0***	***0***	***0***	***0***	***1***	***0***
0	***0***	***1***	***0***	***0***	***0***	***0***	***0***	***1***	***1***
0	***0***	***0***	***1***	***0***	***0***	***0***	***1***	***0***	***0***
0	***0***	***0***	***0***	***1***	***0***	***0***	***1***	***0***	***1***
0	***0***	***0***	***0***	***0***	***1***	***0***	***1***	***1***	***0***
0	***0***	***0***	***0***	***0***	***0***	***1***	***1***	***1***	***1***

LOGIC DIAGRAM FOR DECODER:



TRUTH TABLE:

INPUT			OUTPUT			
E	A	B	D0	D1	D2	D3
1	0	0	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

PROCEDURE:

(i) Connections are given as per circuit diagram.

(ii) Logical inputs are given as per circuit diagram.

(iii) Observe the output and verify the truth table.

RESULT:

EXPT NO.:

DATE:

CONSTRUCTION AND VERIFICATION OF 4 BIT RIPPLE COUNTER AND MOD 10/MOD 12 RIPPLE COUNTER

AIM:

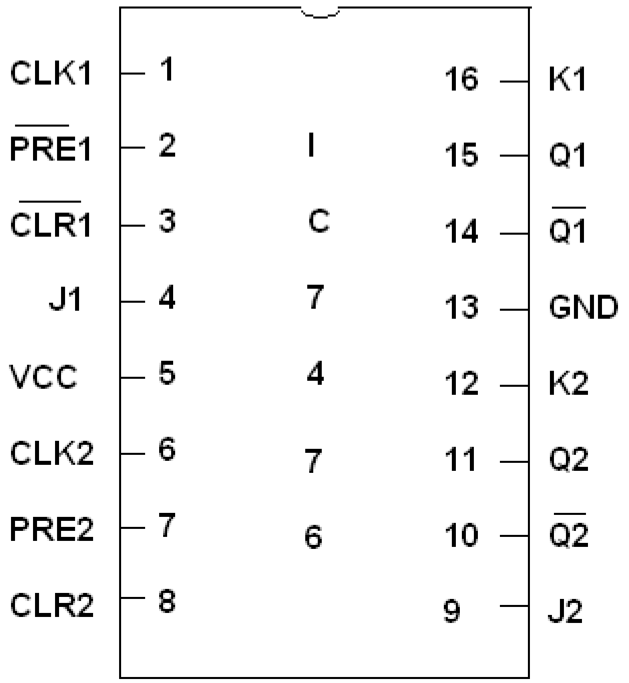
To design and verify 4 bit ripple counter mod 10/ mod 12 ripple counter.

APPARATUS REQUIRED:

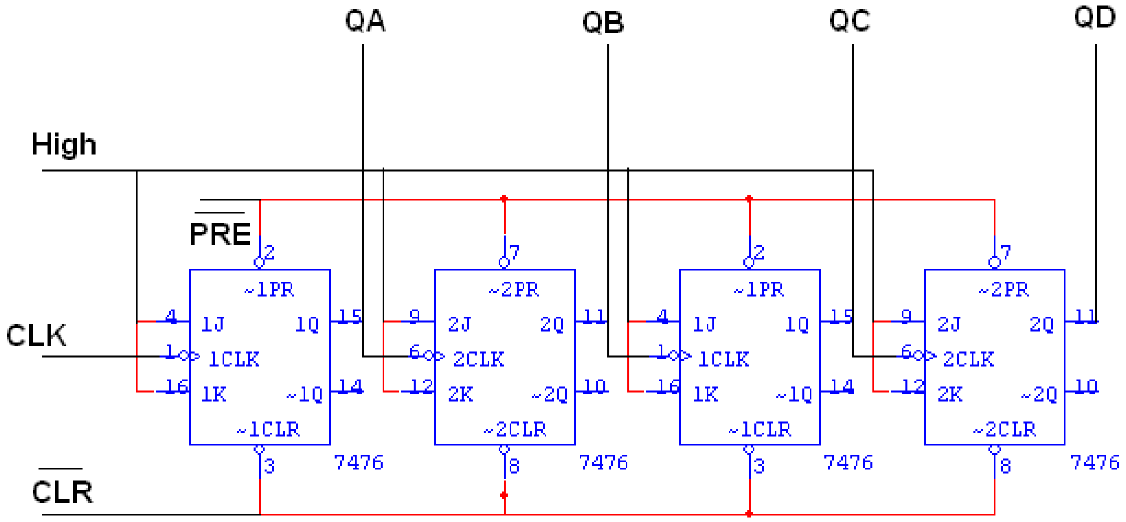
Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	JK FLIP FLOP	IC 7476	2
2.	NAND GATE	IC 7400	1
3.	IC TRAINER KIT	-	1
4.	PATCH CORDS	-	30

THEORY:

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 counter, 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. A soon the clock of second 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.* ***PIN DIAGRAM FOR IC 7476:***



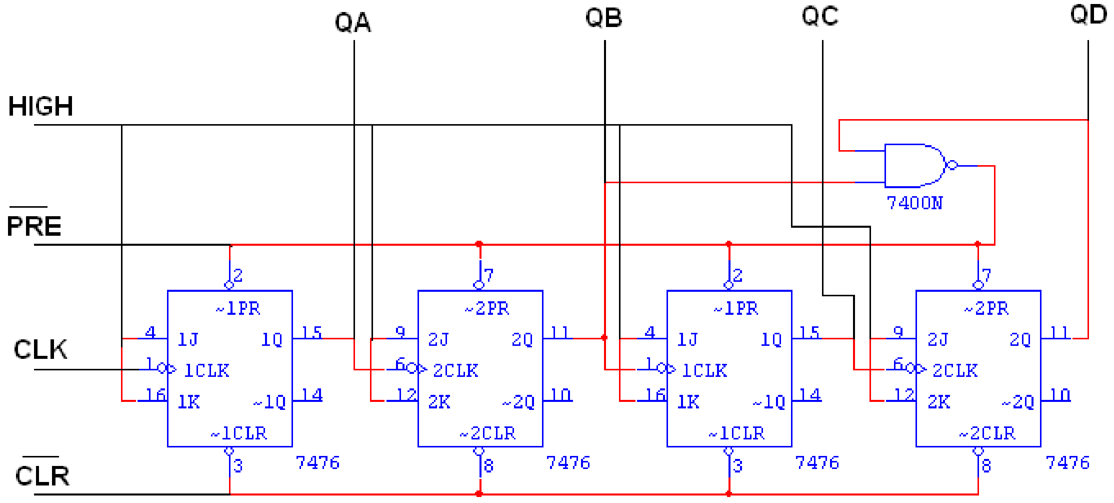
LOGIC DIAGRAM FOR 4 BIT RIPPLE COUNTER:



TRUTH TABLE:

CLK	QA	QB	QC	QD
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1
10	0	1	0	1
11	1	1	0	1
12	0	0	1	1
13	1	0	1	1
14	0	1	1	1
15	1	1	1	1

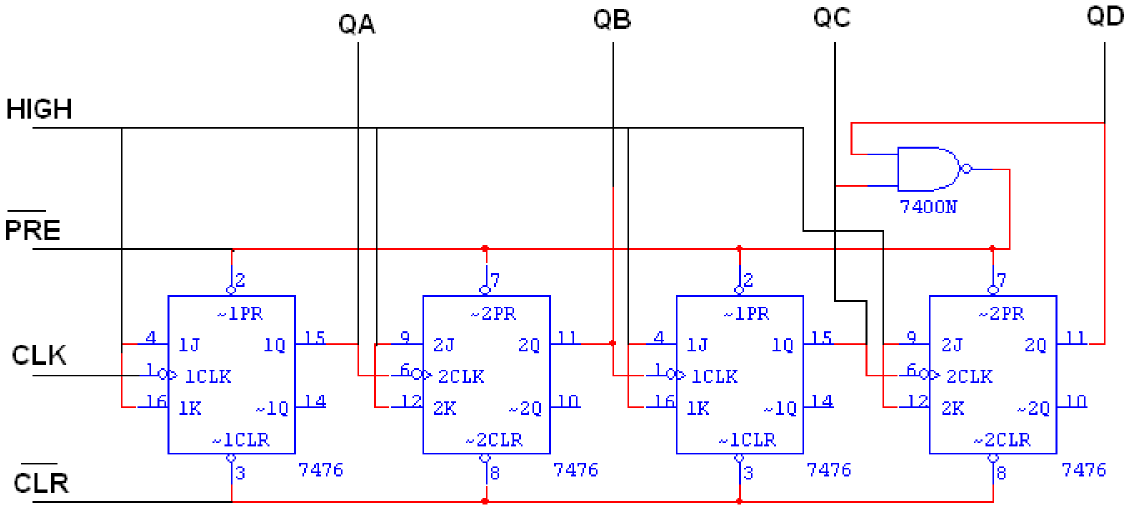
LOGIC DIAGRAM FOR MOD - 10 RIPPLE COUNTER:



TRUTH TABLE:

CLK	*QA*	*QB*	*QC*	*QD*
0	*0*	*0*	*0*	*0*
1	*1*	*0*	*0*	*0*
2	*0*	*1*	*0*	*0*
3	*1*	*1*	*0*	*0*
4	*0*	*0*	*1*	*0*
5	*1*	*0*	*1*	*0*
6	*0*	*1*	*1*	*0*
7	*1*	*1*	*1*	*0*
8	*0*	*0*	*0*	*1*
9	*1*	*0*	*0*	*1*
10	*0*	*0*	*0*	*0*

LOGIC DIAGRAM FOR MOD - 12 RIPPLE COUNTER:



TRUTH TABLE:

CLK	QA	QB	QC	QD
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1
10	0	1	0	1
11	1	1	0	1
12	0	0	0	0

PROCEDURE:

(i) Connections are given as per circuit diagram.

(ii) Logical inputs are given as per circuit diagram.

(iii) Observe the output and verify the truth table.

RESULT:

EXPT NO.:

DATE:

DESIGN AND IMPLEMENTATION OF 3 BIT SYNCHRONOUSUP/DOWN COUNTER

AIM:

To design and implement 3 bit synchronous up/down counter.

APPARATUS REQUIRED:

Sl.No.	*COMPONENT*	*SPECIFICATION*	*QTY.*
1.	*JK FLIP FLOP*	*IC 7476*	*2*
2.	*3 I/P AND GATE*	*IC 7411*	*1*
3.	*OR GATE*	*IC 7432*	*1*
4.	*XOR GATE*	*IC 7486*	*1*
5.	*NOT GATE*	*IC 7404*	*1*
6.	*IC TRAINER KIT*	*-*	*1*
7.	*PATCH CORDS*	*-*	*35*

THEORY:

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. An up/down counter is one that is capable of progressing in increasing order or decreasing order through a certain sequence. An up/down counter is also called bidirectional counter. Usually up/down operation of the counter is controlled by up/down signal. When this signal is high counter goes through up sequence and when up/down signal is low counter follows reverse sequence.

K MAP

QB QC
UD QA

1	0	0	0
X	X	X	X
X	X	X	X
0	0	1	0

JA = $\overline{UD}\ \overline{QB}\ \overline{QC} + UD\ QB\ QC$

QB QC
UD QA

X	X	X	X
1	0	0	0
0	0	1	0
X	X	X	X

KA = $\overline{UD}\ \overline{QB}\ \overline{QC} + UD\ QB\ QC$

QB QC
UD QA

1	X	X	1
1	X	X	1
1	X	X	1
1	X	X	1

JC = 1

QB QC
UD QA

1	0	X	X
1	0	X	X
0	1	X	X
0	1	X	X

JB = $\overline{UD \oplus QC}$

QB QC
UD QA

X	X	0	1
X	X	0	1
X	X	1	0
X	X	1	0

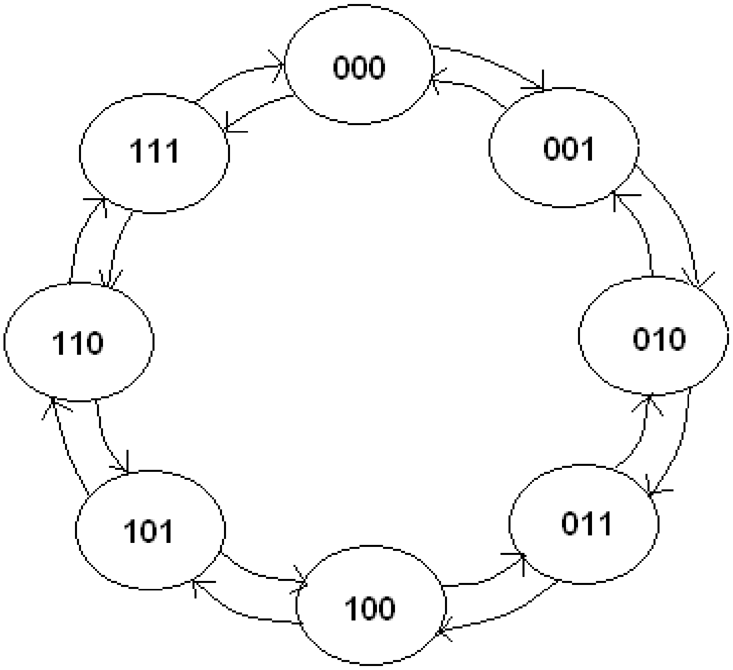
KB = $(\overline{UD \oplus QC})$

QB QC
UD QA

X	1	1	X
X	1	1	X
X	1	1	X
X	1	1	X

KC = 1

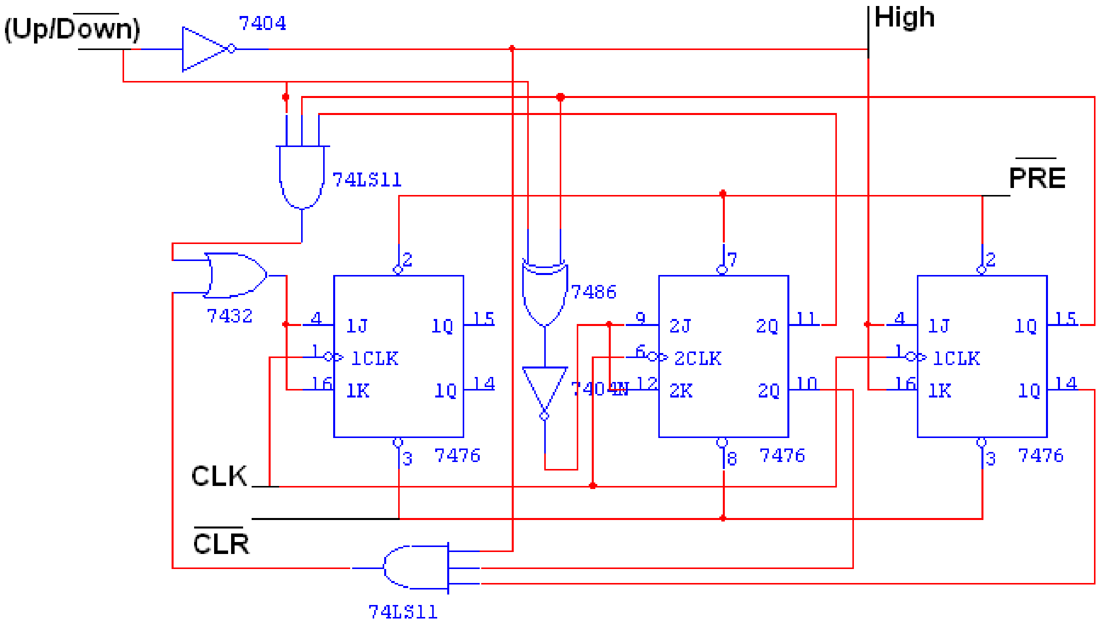
STATE DIAGRAM:



CHARACTERISTICS TABLE:

Q	Q_{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

LOGIC DIAGRAM:



TRUTH TABLE:

Input Up/Down	*Present State*			*Next State*			*A*		*B*		*C*	
	Q_A	Q_B	Q_C	Q_{A+1}	Q_{B+1}	Q_{C+1}	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	1	1	1	1	X	1	X	1	X
0	1	1	1	1	1	0	X	0	X	0	X	1
0	1	1	0	1	0	1	X	0	X	1	1	X
0	1	0	1	1	0	0	X	0	0	X	X	1
0	1	0	0	0	1	1	X	1	1	X	1	X
0	0	1	1	0	1	0	0	X	X	0	X	1
0	0	1	0	0	0	1	0	X	X	1	1	X
0	0	0	1	0	0	0	0	X	0	X	X	1
1	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
1	0	1	0	0	1	1	0	X	X	0	1	X
1	0	1	1	1	0	0	1	X	X	1	X	1
1	1	0	0	1	0	1	X	0	0	X	1	X
1	1	0	1	1	1	0	X	0	1	X	X	1
1	1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	1	0	0	0	X	1	X	1	X	1

PROCEDURE:

(i) Connections are given as per circuit diagram.

(ii) Logical inputs are given as per circuit diagram.

(iii) Observe the output and verify the truth table.

RESULT:

EXPT NO.

:DATE:

DESIGN AND IMPLEMENTATION OF SHIFT REGISTER

AIM

: *To design and implement*

- *(i) Serial in serial out*
- *(ii) Serial in parallel out*
- *(iii) Parallel in serial out*
- *(iv) Parallel in parallel out*

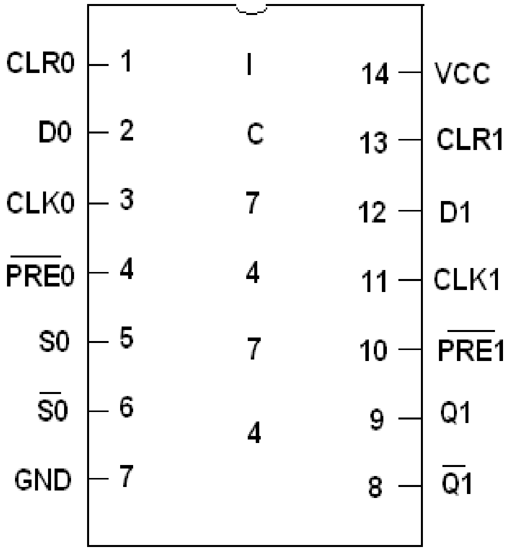
APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	D FLIP FLOP	IC 7474	2
2.	OR GATE	IC 7432	1
3.	IC TRAINER KIT	-	1
4.	PATCH CORDS	-	35

THEORY:

A register is capable of shifting its binary information in one or both directions is known as shift register. The logical configuration ofshift register consist of a D-Flip flop cascaded with output of one flipflop connected to input of next flip flop. All flip flops receive commonclock pulses which causes the shift in the output of the flip flop.The simplest possible shift register is one that uses only flip flop. Theoutput of a given flip flop is connected to the input of next flip flop of *the register. Each clock pulse shifts the content of register one bit position to right.*

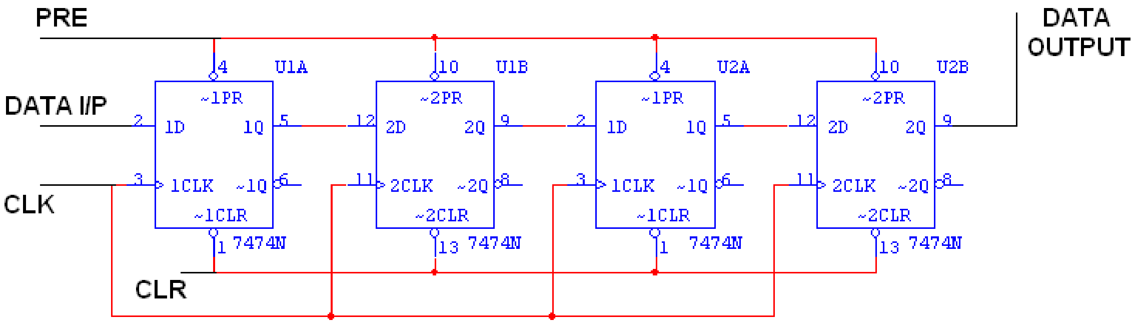
PIN DIAGRAM:



LOGIC DIAGRAM:

SERIAL IN SERIAL

OUT:

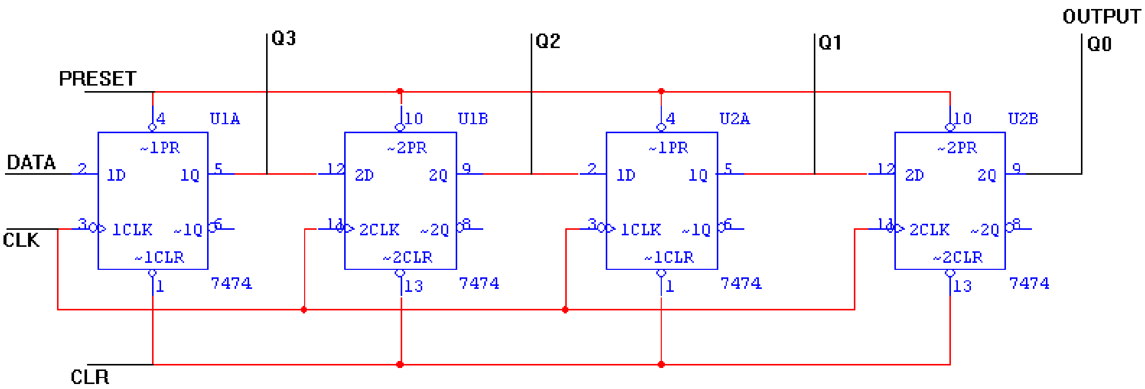


TRUTH TABLE:

CLK	Serial in	Serial out
1	1	0
2	0	0
3	0	0
4	1	1
5	X	0
6	X	0
7	X	1

LOGIC DIAGRAM:

SERIAL IN PARALLEL OUT:



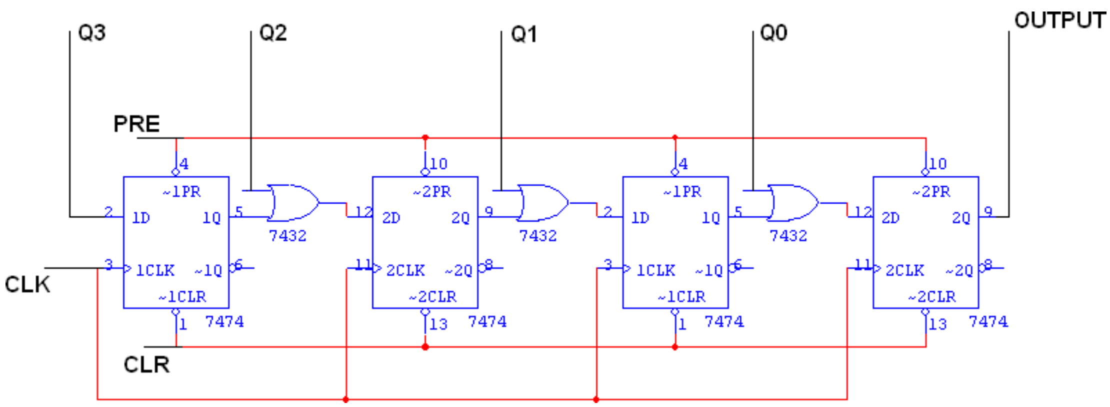
TRUTH TABLE:

CLK	DATA	OUTPUT			
		Q_A	Q_B	Q_C	Q_D
1	1	1	0	0	0
2	0	0	1	0	0

3	*0*	*0*	*0*	*1*	*1*
4	*1*	*1*	*0*	*0*	*1*

LOGIC DIAGRAM:

PARALLEL IN SERIAL OUT:

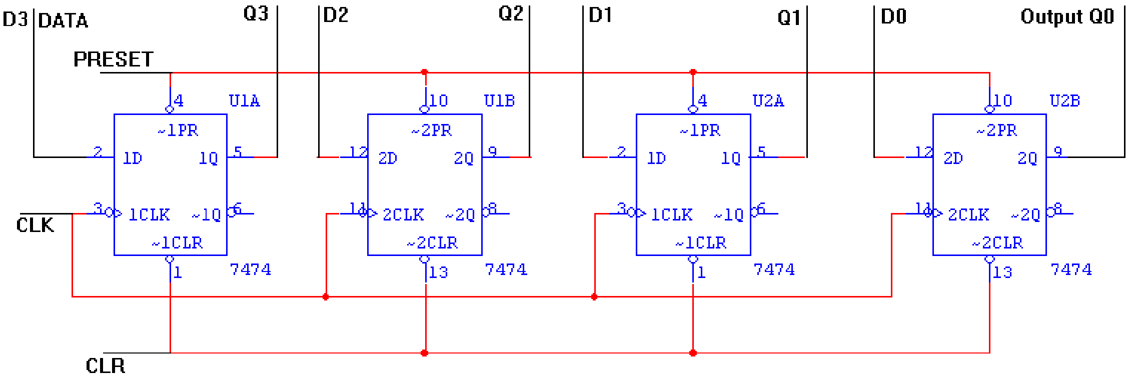


TRUTH TABLE:

CLK	*Q3*	*Q2*	*Q1*	*Q0*	*O/P*
0	*1*	*0*	*0*	*1*	*1*
1	*0*	*0*	*0*	*0*	*0*
2	*0*	*0*	*0*	*0*	*0*
3	*0*	*0*	*0*	*0*	*1*

LOGIC DIAGRAM:

PARALLEL IN PARALLEL OUT:



TRUTH TABLE:

CLK	***DATA INPUT***				***OUTPUT***			
	D ***A***	***D*** ***B***	***D*** ***C***	***D*** ***D***	***Q*** ***A***	***Q*** ***B***	***Q*** ***C***	***Q*** ***D***
1	***1***	***0***	***0***	***1***	***1***	***0***	***0***	***1***
2	***1***	***0***	***1***	***0***	***1***	***0***	***1***	***0***

PROCEDURE:

(i) *Connections are given as per circuit diagram.*

(ii) *Logical inputs are given as per circuit diagram.*

(iii) *Observe the output and verify the truth table.*

RESULT: