SAVEETHA ENGINEERING COLLEGE

AUTONOMOUS

NAAC A GRADE

Accredited by NBA NATIONAL BOARD OF ACCREDITATION 5 UG Programs

Affiliated to Anna University | Approved by AICTE

DEPARTMENT OF ELECTRONICS AND COMMUNICATION

ENGINEERING

19EC303 DIGITAL PRINCIPLES AND SYSTEM DESIGN

LAB MANUAL

COMMON FOR ECE, CSE, BME, ME AND IT

19EC303 DIGITAL PRINCIPLES AND SYSTEM DESIGN

LAB MANUAL

<u>SYLLABUS</u>

1. Study of logic gates
2.Verification of Boolean Laws and theorems

3. Design and implementation of Adders and Subtractors using logic gates.

4. Design and implementation of code converters using logic gates

    a. BCD to excess-3 code and vice versa

    b. Binary to gray and vice-versa

5. Design and implementation of 4 bit binary Adder/ subtractor using IC 7483

6. Design and implementation of BCD adder using IC 7483

7.Design and implementation of 2 Bit Magnitude Comparator using logic gates 8 Bit

Magnitude Comparator using IC 7485

8. Design and implementation of Multiplexer and De-multiplexer using logic gates
9.. Design and implementation of encoder ,Priority encoder and decoder using logic gates
10.Verification of Flip Flops using Basic gates.
11. Construction and verification of 4 bit ripple counter and Mod-10 / Mod-12 Ripple Counters
12.Design and implementation of 3-bit ring counter
13. Coding combinational circuits using HDL.
14. Coding sequential circuits using HDL.

INDEX

EXP.NO	DATE	TITLE OF EXPERIMENT	PAGE NO	MARKS	STAFF INITIAL
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					

VERIFICATION OF LOGIC GATES

EXP NO. : 1

DATE :

AIM

To verify the logic gates operation with its 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	-	As per Requirement

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 & NOT are basic gates. NAND, NOR are known as universal gates. Basic gates can be formed from these gates. XOR & XNOR are derived 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 inputs is low. The output is low level when both inputs are high.

NOR GATE

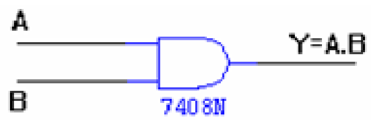
The NOR gate is a contraction of OR-NOT. The output is high when both inputs are low. The output is low when one or both inputs are high.

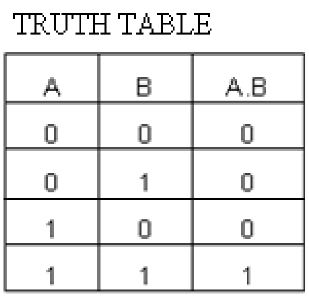
X-OR GATE

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.

AND GATE

SYMBOL

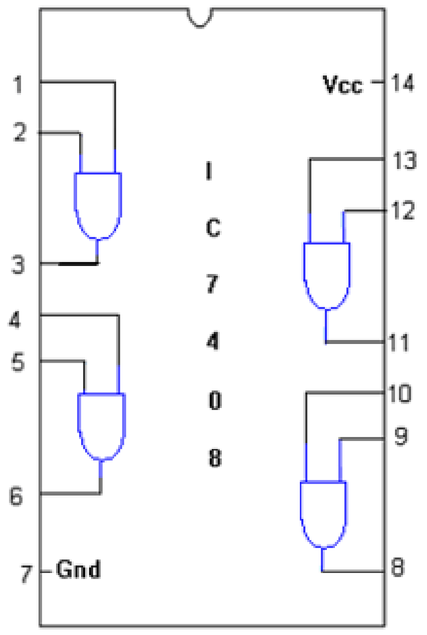




TRUTH TABLE

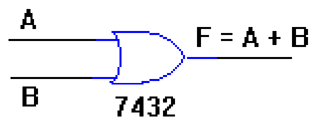
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

PIN DIAGRAM

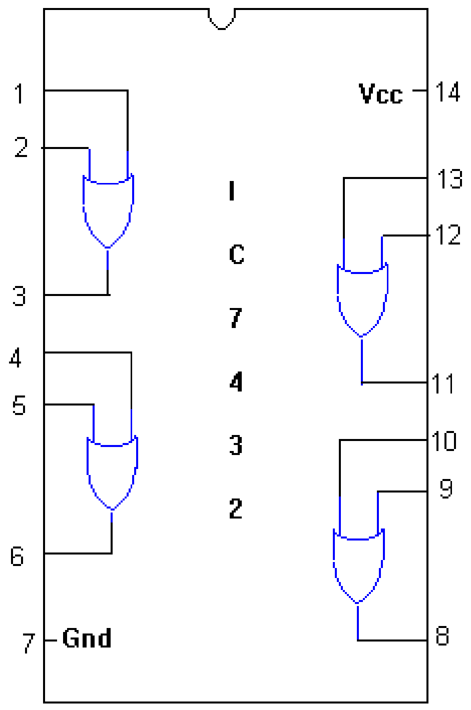


OR GATE

SYMBOL



PIN DIAGRAM

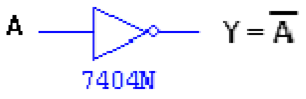


TRUTH TABLE

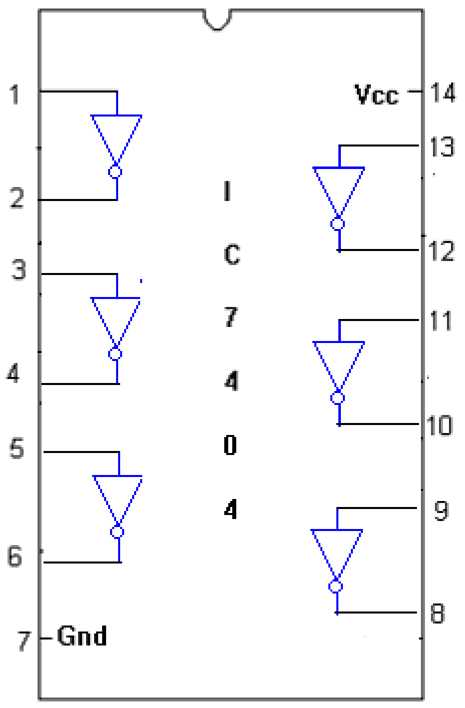
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

NOT GATE

SYMBOL



PIN DIAGRAM

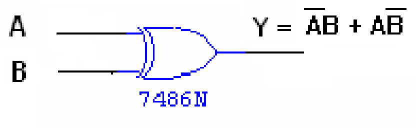


TRUTH TABLE

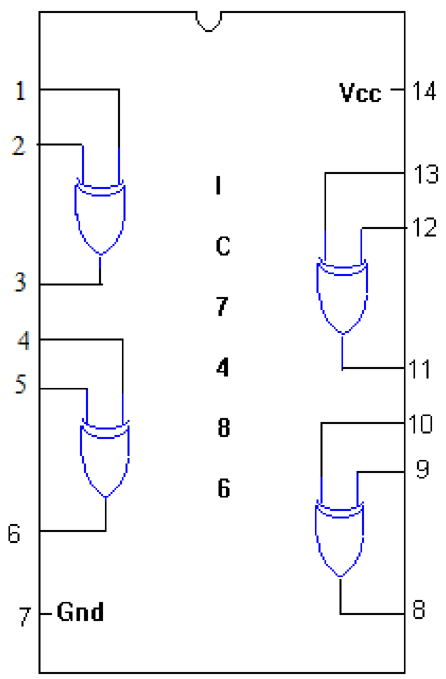
A	$\overline{A}$
0	1
1	0

EX-OR GATE

SYMBOL



PIN DIAGRAM

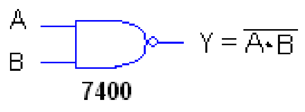


TRUTH TABLE

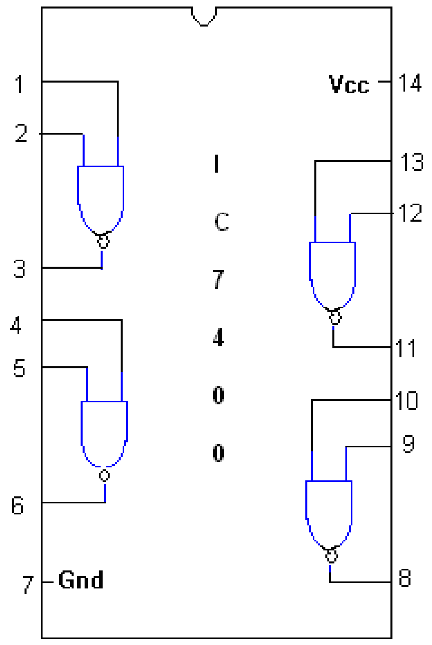
A	B	$\overline{A}B + A\overline{B}$
0	0	0
0	1	1
1	0	1
1	1	0

2-INPUT NAND GATE

SYMBOL



PIN DIAGRAM

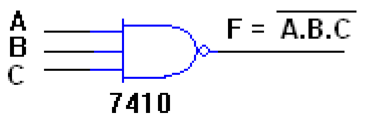


TRUTH TABLE

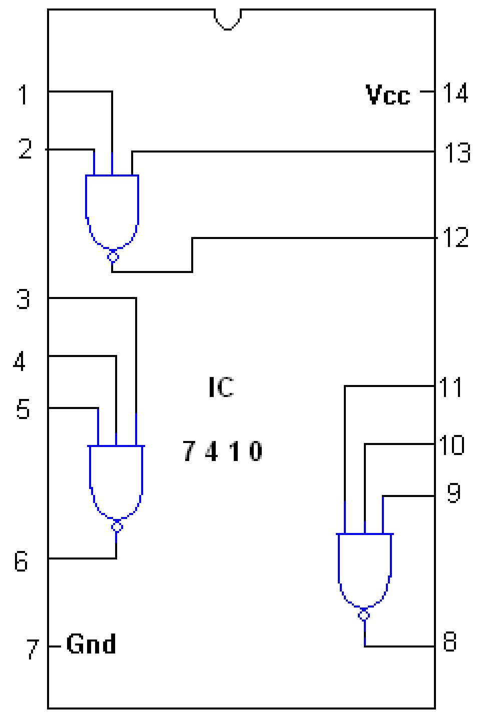
A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

3-INPUT NAND GATE

SYMBOL



PIN DIAGRAM

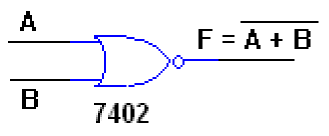


TRUTH TABLE

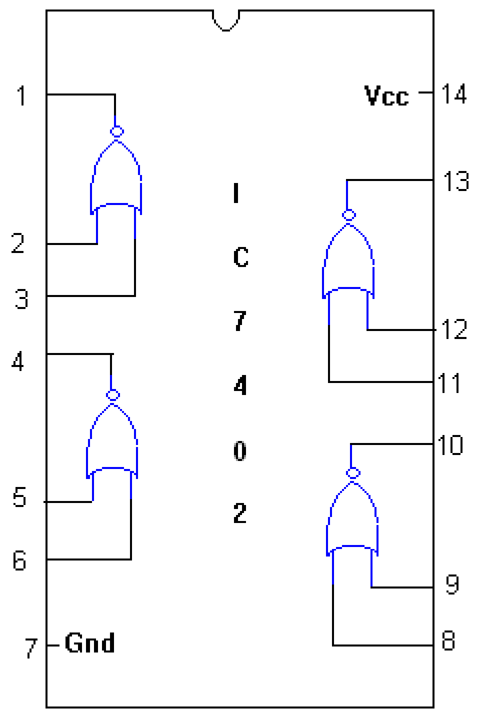
A	B	C	$\overline{A.B.C}$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

NOR GATE

SYMBOL



PIN DIAGRAM



TRUTH TABLE

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

PROCEDURE

Connections are given as per circuit diagram.

Logical inputs are given as per circuit diagram.

Observe the output and verify the truth table.

RESULT :

Thus the working of the logic gates was studied and their truth tables were verified.

VERIFICATION OF BOOLEAN THEOREMS

EXP NO. : 2
DATE :

AIM:

To study and verify the Boolean theorems using logic gates.

COMPONENTS REQUIRED:

S.No.	Apparatus	Specifications	Quantity
1.	IC Trainer kit	--	1 no
2.	Logic gate IC's	IC 7404, IC 7408	1no each
3.	Logic gate IC's	IC 7402, IC 7486	1no each
4.	Connecting wires	--	1 set

Theorems:

1. **Idempotence laws:**
   a) x+x =x
   b) x.x=x

2. **Identity law:**
   x+1 = 1

3. **Null law:**
   x.0 = 0

4. **Involution law (or) double negation law:**
   (x ′) ′ = x

5. **Associative law:**
   x+(y+z) = (x+y)+z
   x.(y.z) = (x.y).z

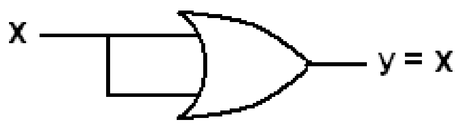
6. **Demorgan's law:**
   (x+y)′ = x′ . y′
   (x.y)′ = x′ + y′

7. **Adsorption theorem:**
   x+(x.y) = x
   x.(x+y) = x

1. **Idempotence laws:** **TRUTH TABLE**

   a) x+x =x

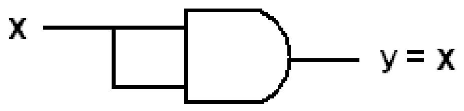
x	x+x=x
0	0
1	1



IC 7432

b) x.x=x

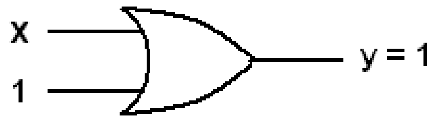
IC 7408



x	x.x=x
0	0
1	1

2. Identity law:
x+1 = 1

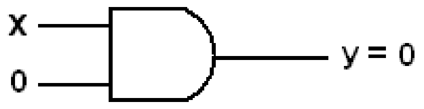
IC 7432



x	x+1=1
0	1
1	1

3. Null law:
x.0 = 0

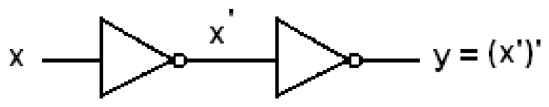
IC 7408



x	x.0=0
0	0
1	0

4. Involution law (or) double negation law:
(x ′) ′ = x

IC 7404 IC 7404



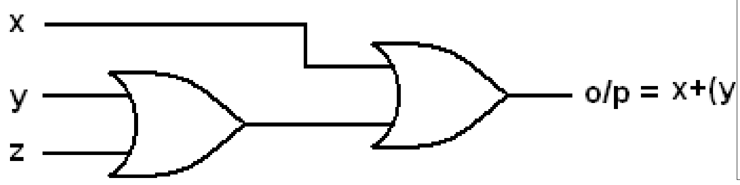
x	x′	(x′)′=x
0	1	0
1	0	1

5. Associative law:

a) x+(y+z) = (x+y)+z

x	y	z	y+z	x+ (y+z)	x+y	(x+y) +z
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

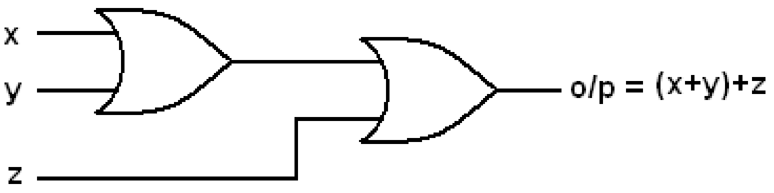
L.H.S



IC 7432 **IC 7432**

R.H.S

IC 7432 **IC 7432**

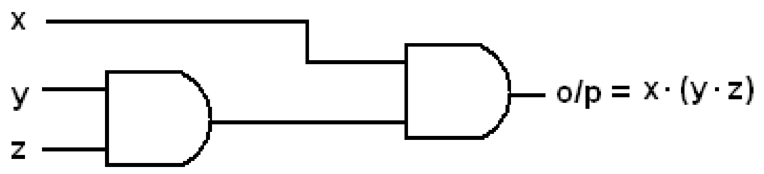


b) x.(y.z) = (x.y).z

L.H.S

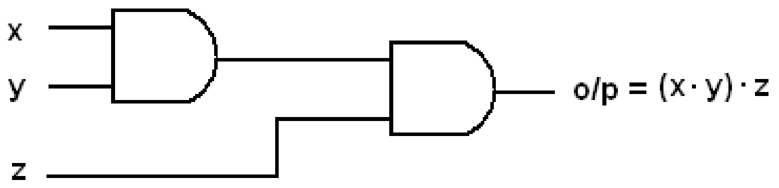
x	y	z	y.z	x. (y.z)	x.y	(x.y) .z
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	0	1	0
1	1	1	1	1	1	1

IC 7408 **IC 7408**



R.H.S

IC 7408 **IC 7408**

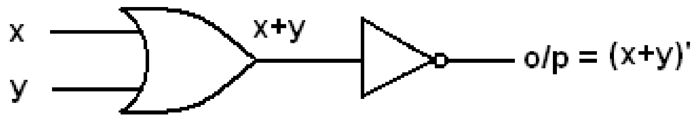


6. **Demorgan’s law:**

a) **(x+y)´ = x´ . y´**

L.H.S

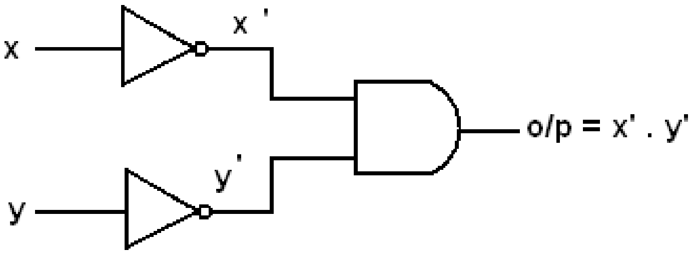
IC 7432 **IC 7404**



x	Y	x+y	(x+y) ´	x´	y´	x´. y´
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

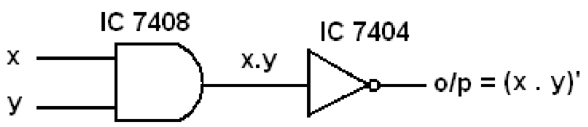
R.H.S

IC 7404 **IC 7408**



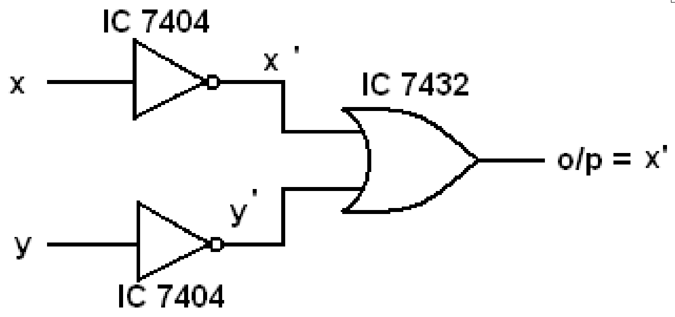
b) **(x.y)´ = x´ + y´**

L.H.S



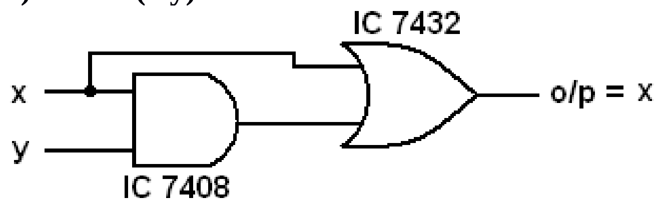
x	Y	x.y	(x.y) ´	x´	y´	x´+y´
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

R.H.S



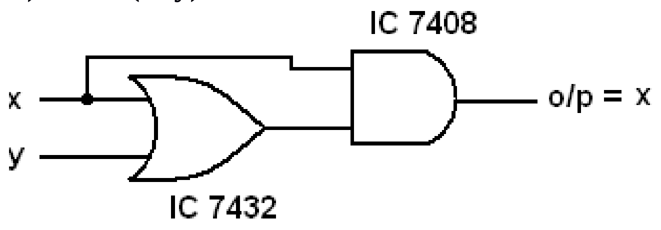
7. **Adsorption theorem:**

a) **x+(x.y) = x**



X	y	x.y	x+(x.y)=x
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

b) **x.(x+y) = x**



X	y	x+y	x.(x+y)=x
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

Procedure:

1. **Connections are made as per the circuit diagram for each of the theorems.**
2. **Switch on the IC trainer kit.**
3. **Apply logic inputs 0 or 1 to input variables**
4. **Verify the truth table by observing the output indicators for all the theorems.**

Result:

Thus, the Boolean theorems and Laws are studied and verified using logic gates.

DESIGN OF ADDER AND SUBTRACTOR

EXP NO. : 3

DATE :

AIM

To design and construct half adder, full adder, half subtractor and full subtractor circuits and verify the truth table using 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	-	As per

			Requirement

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.

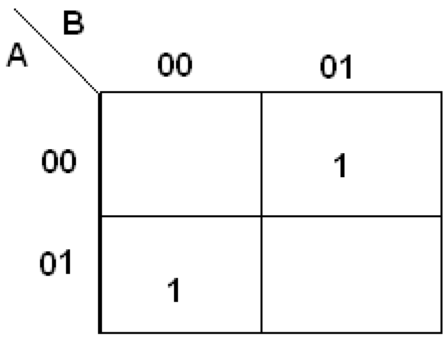
HALF ADDER

TRUTH TABLE

A	B	CARRY	SUM

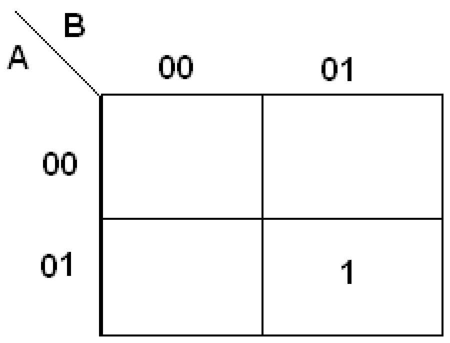
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

K-MAP FOR SUM



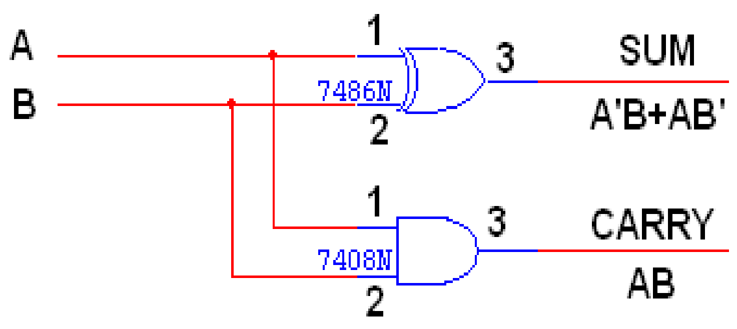
SUM = A’B + AB’

K-MAP FOR CARRY



CARRY = AB

LOGIC DIAGRAM



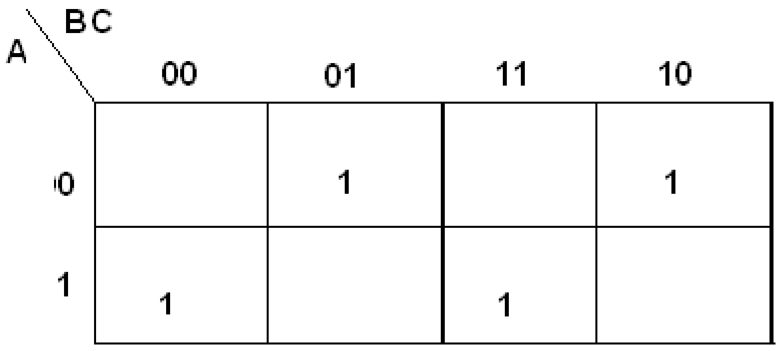
FULL ADDER

TRUTH TABLE

A	B	C	CARRY	SUM
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1

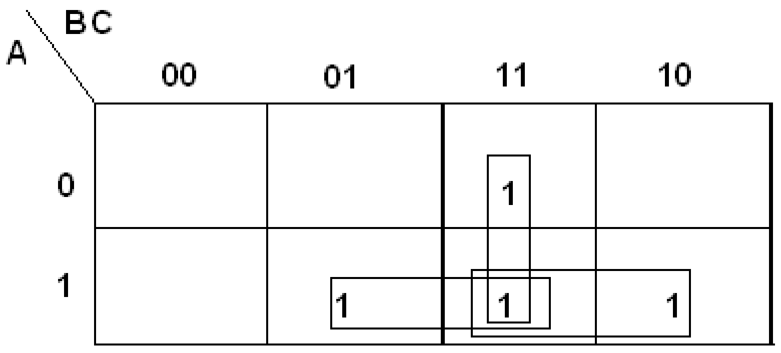
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

K-MAP FOR SUM



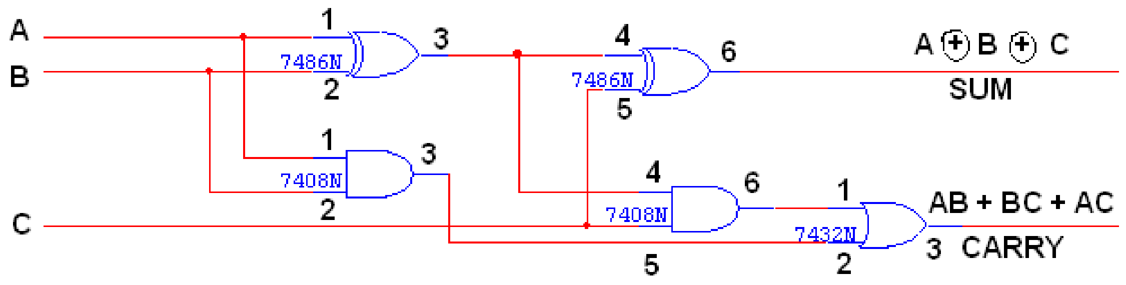
SUM = A’B’C + A’BC’ + ABC’ + ABC

K-MAP FOR CARRY



CARRY = AB + BC + AC

LOGIC DIAGRAM

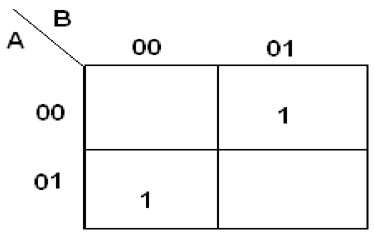


HALF SUBTRACTOR

TRUTH TABLE

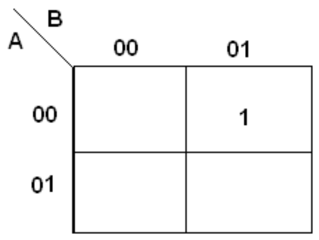
A	B	BORROW	DIFFERENCE
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

K-MAP FOR DIFFERENCE



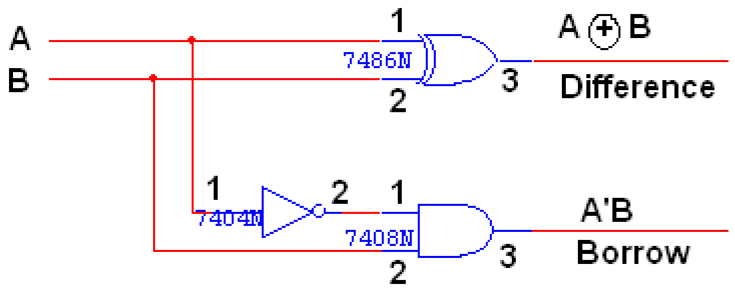
DIFFERENCE = A'B + AB'

K-MAP FOR BORROW



BORROW = A'B

LOGIC DIAGRAM

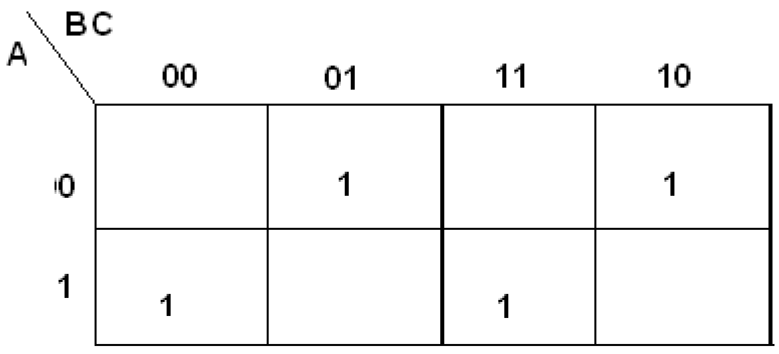


FULL SUBTRACTOR

TRUTH TABLE

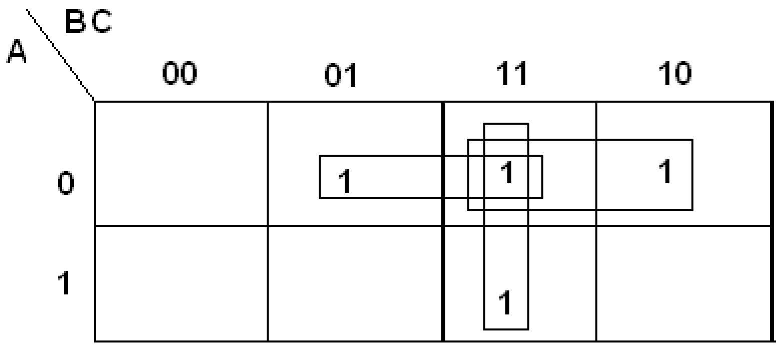
A	B	C	BORROW	DIFFERENCE
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

K-MAP FOR DIFFERENCE



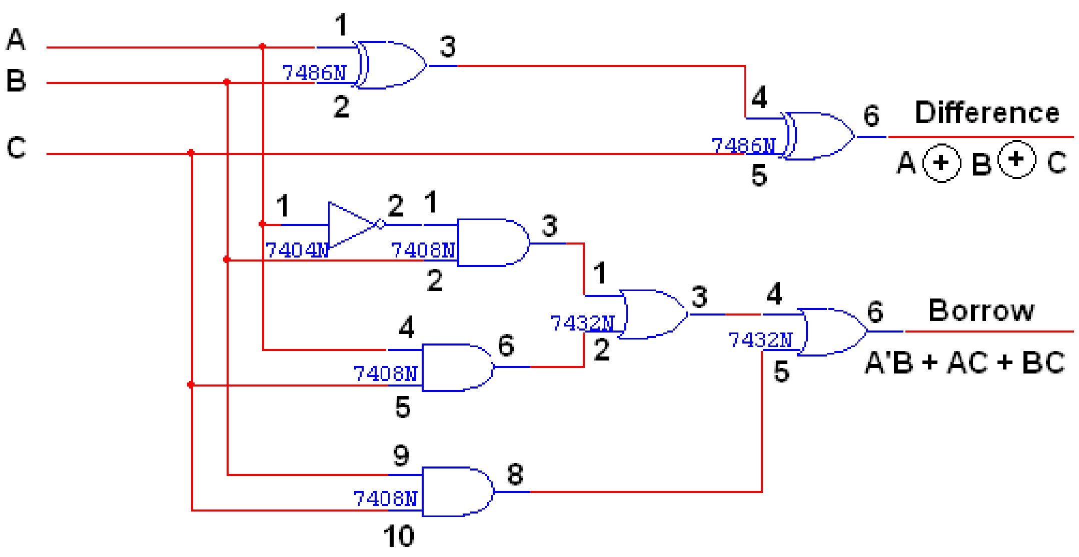
DIFFERENCE = A’B’C + A’BC’ + AB’C’ + ABC

K-MAP FOR BORROW

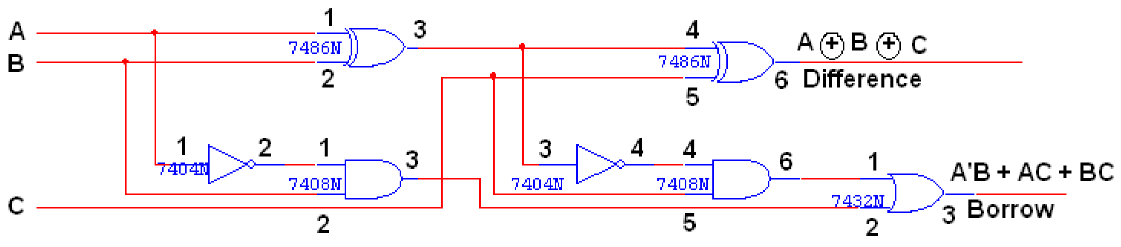


BORROW = A'B + BC + A'C

LOGIC DIAGRAM



FULL SUBTRACTOR USING TWO HALF SUBTRACTOR



PROCEDURE

Connections are given as per circuit diagram.

Logical inputs are given as per circuit diagram.

Observe the output and verify the truth table.

RESULT

Thus, the Adder and Subtractor are studied and verified using logic gates

DESIGN AND IMPLEMENTATION OF CODE CONVERTOR

EXP NO. : 4

DATE :

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	-	As per Requirement

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.

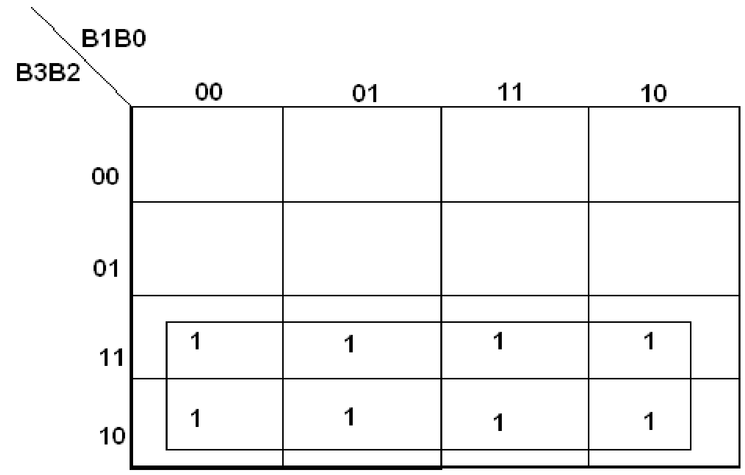
BINARY TO GRAY CODE CONVERTOR

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**

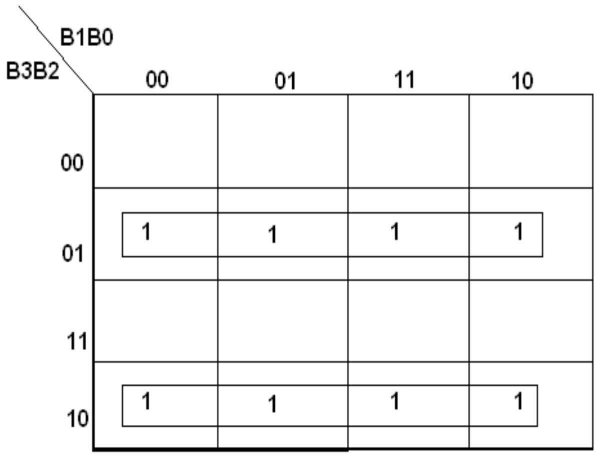
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

K-MAP FOR G_3



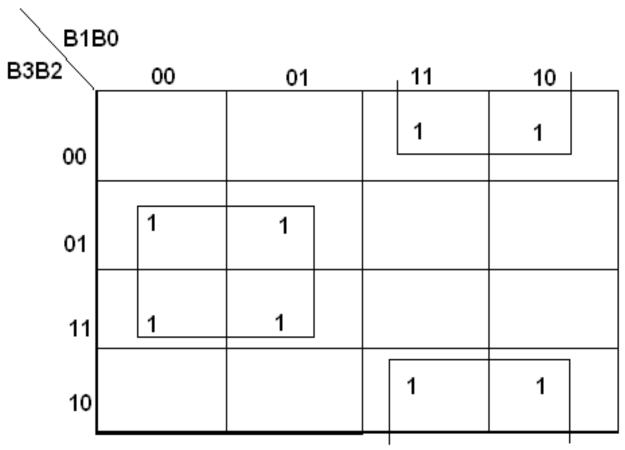
$G_3 = B_3$

K-MAP FOR G_2



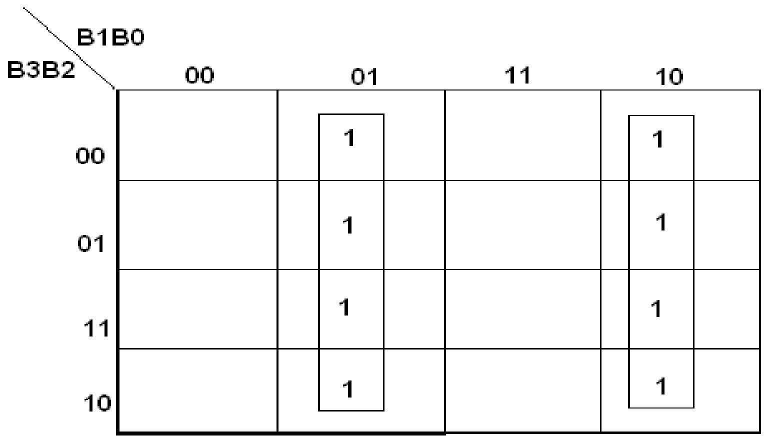
$G2 = B3 \oplus B2$

K-MAP FOR G_1



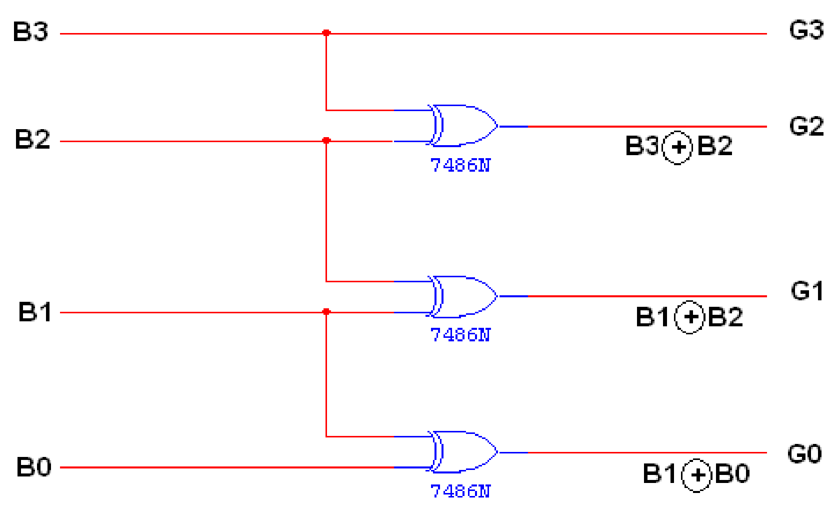
$G1 = B1 \oplus B2$

K-MAP FOR G_0



$G0 = B1 \oplus B0$

LOGIC DIAGRAM

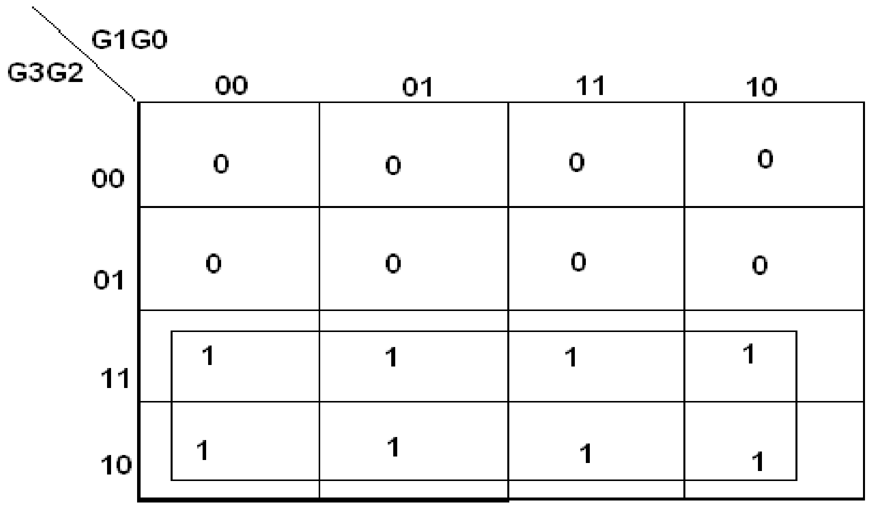


GRAY CODE TO BINARY CONVERTOR

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

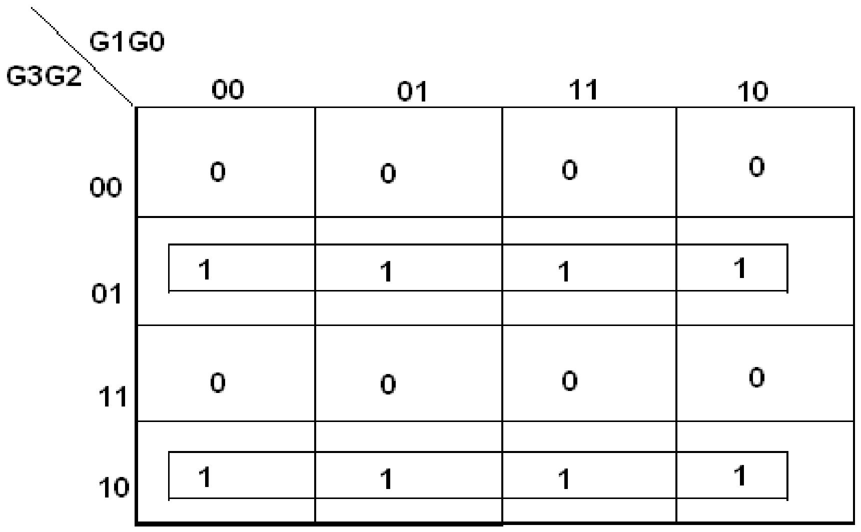
K-MAP FOR B_3



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

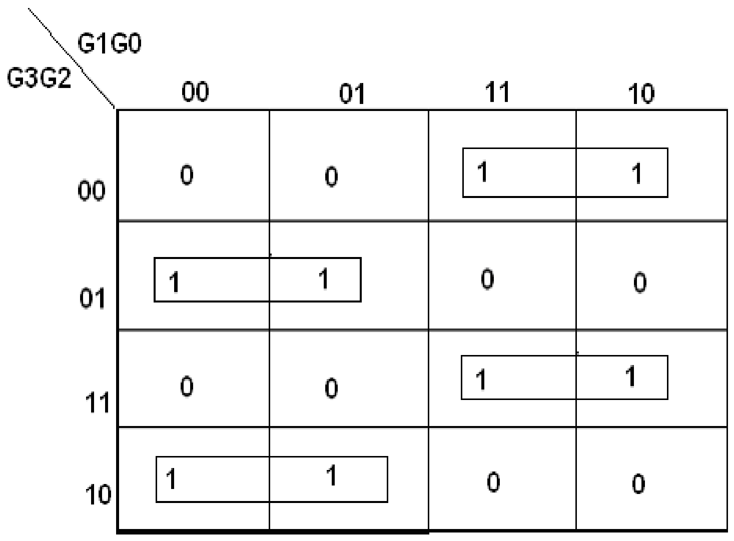
K-MAP FOR B_2



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

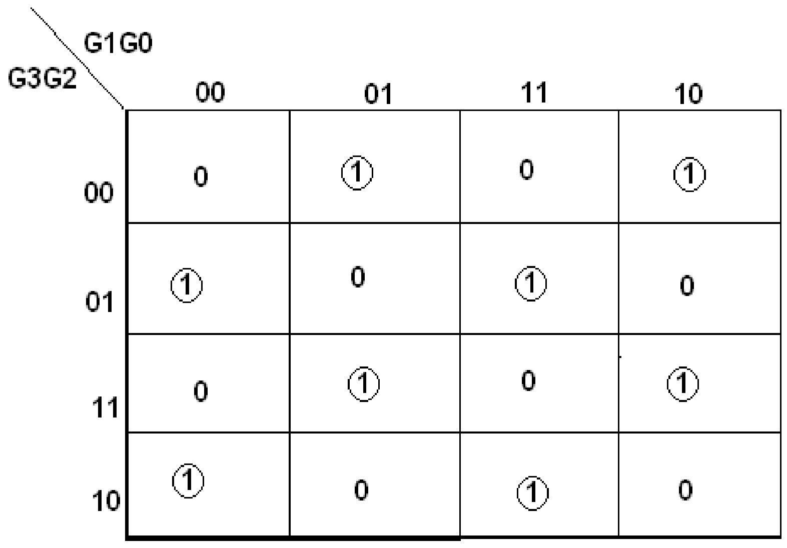
$$B2 = G3 \oplus G2$$

K-MAP FOR B_1



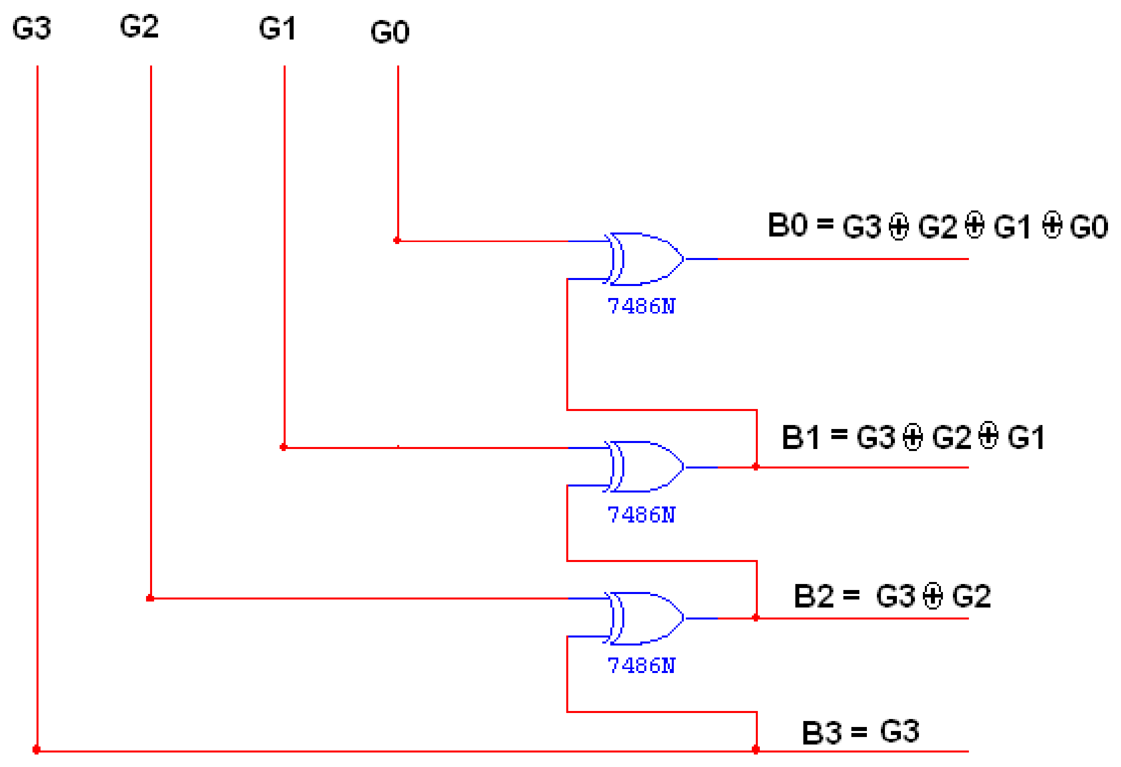
$B1 = G3 \oplus G2 \oplus G1$

K-MAP FOR B_0



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

LOGIC DIAGRAM

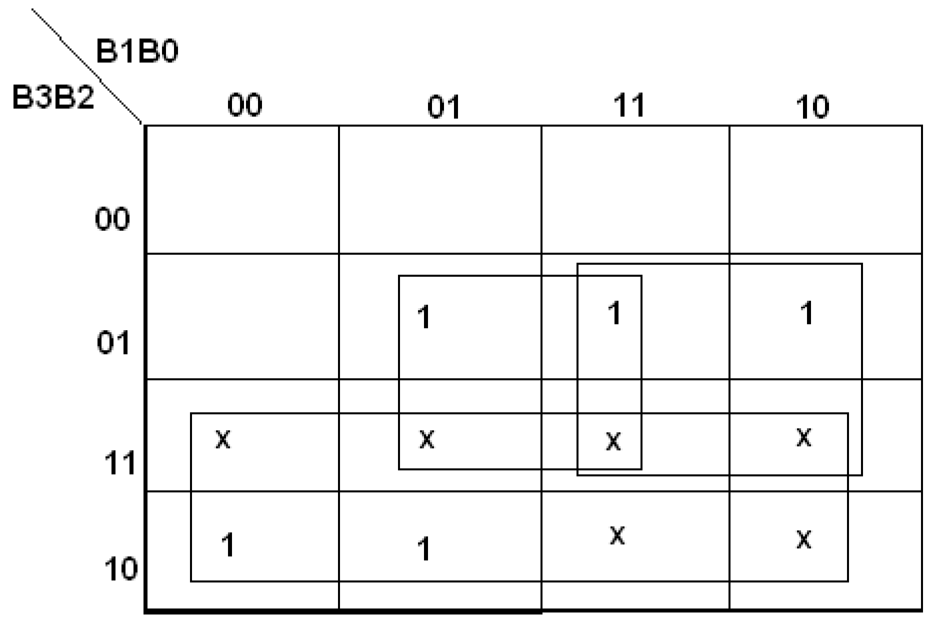


BCD TO EXCESS-3 CONVERTOR

TRUTH TABLE

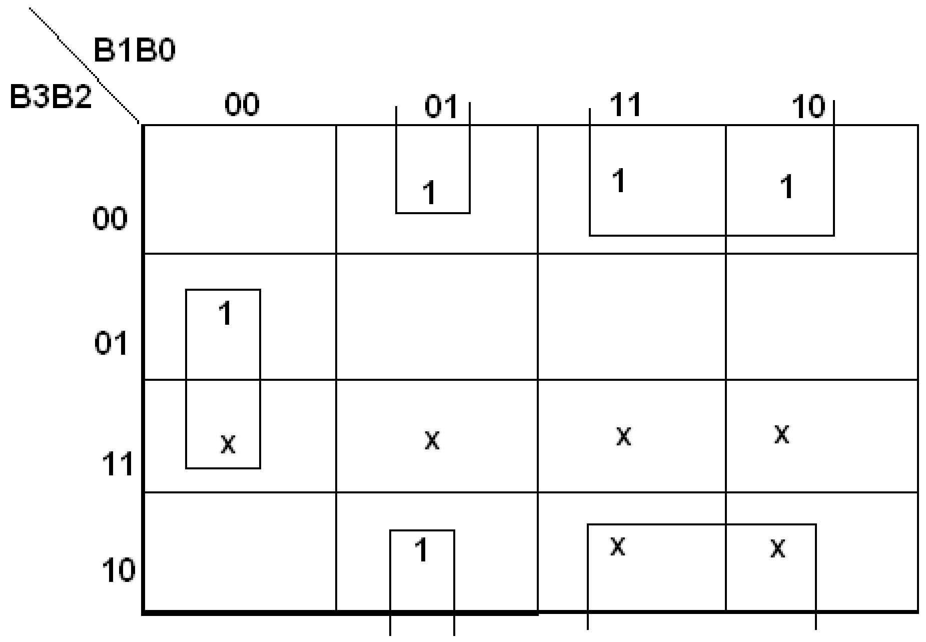
BCD input				Excess – 3 output			
B3	**B2**	**B1**	**B0**	**G3**	**G2**	**G1**	**G0**
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	x	x	x	x
1	0	1	1	x	x	x	x
1	1	0	0	x	x	x	x
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x

K-MAP FOR E_3



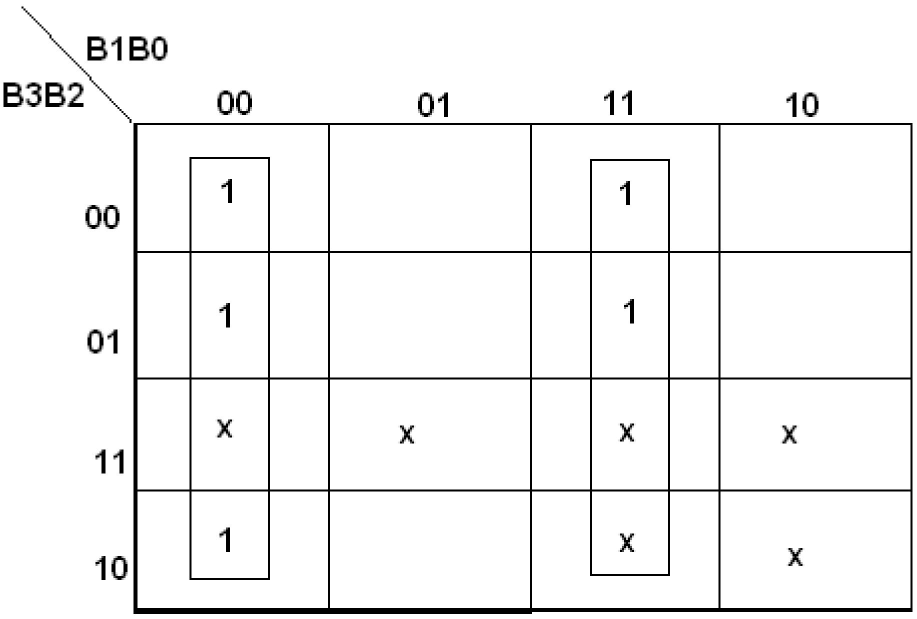
$E3 = B3 + B2\ (B0 + B1)$

K-MAP FOR E_2



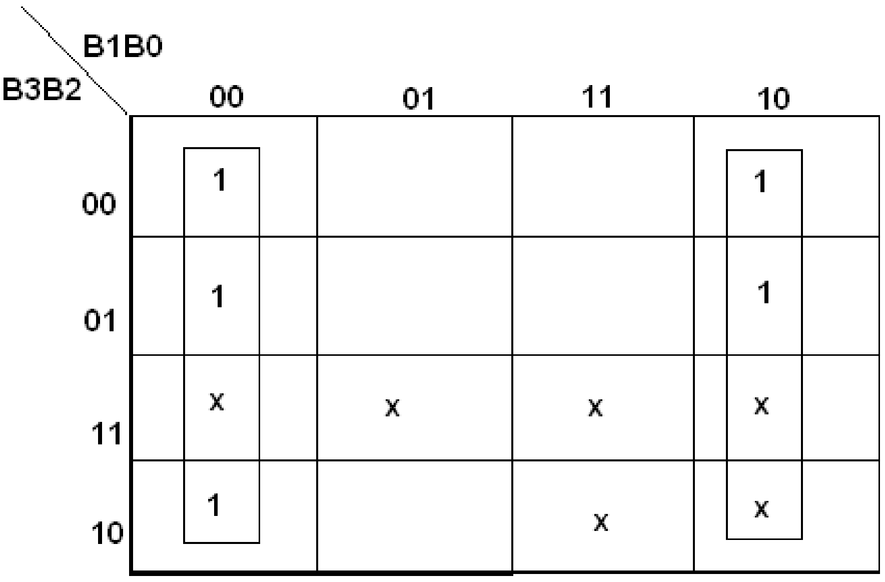
$E2 = B2 \oplus (B1 + B0)$

K-MAP FOR E_1



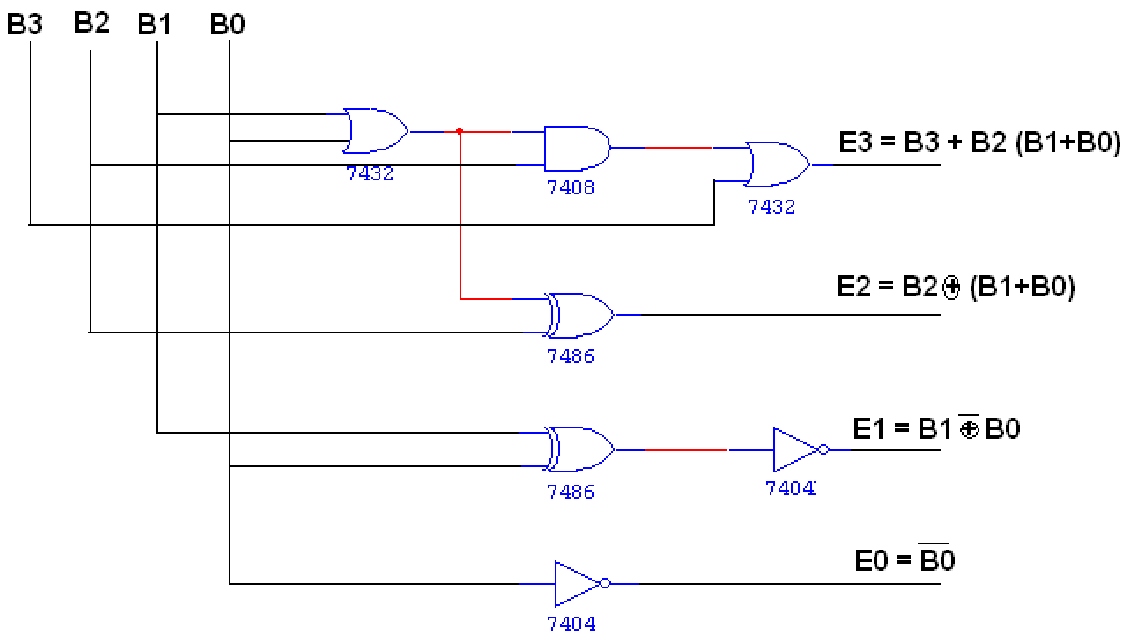
$$E1 = B1 \overline{\oplus} B0$$

K-MAP FOR E_0



$$E0 = \overline{B0}$$

LOGIC DIAGRAM

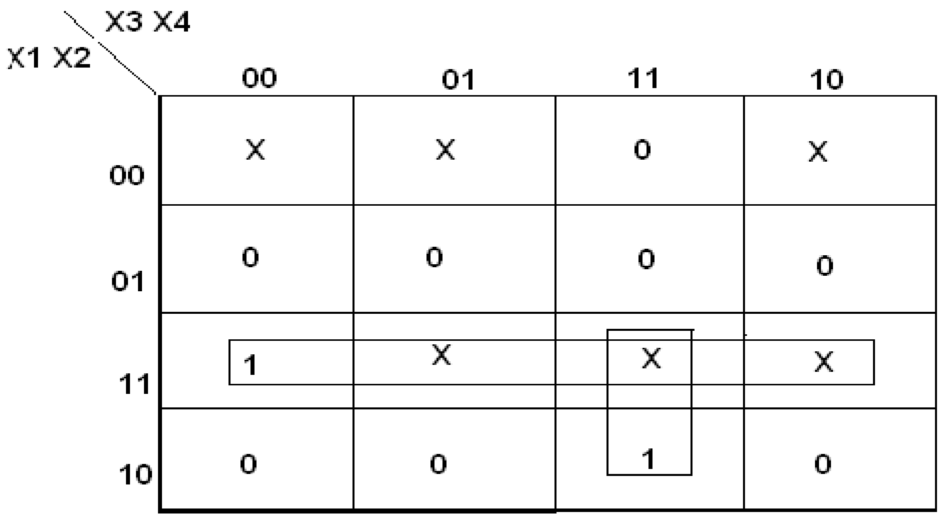


EXCESS-3 TO BCD CONVERTOR

TRUTH TABLE

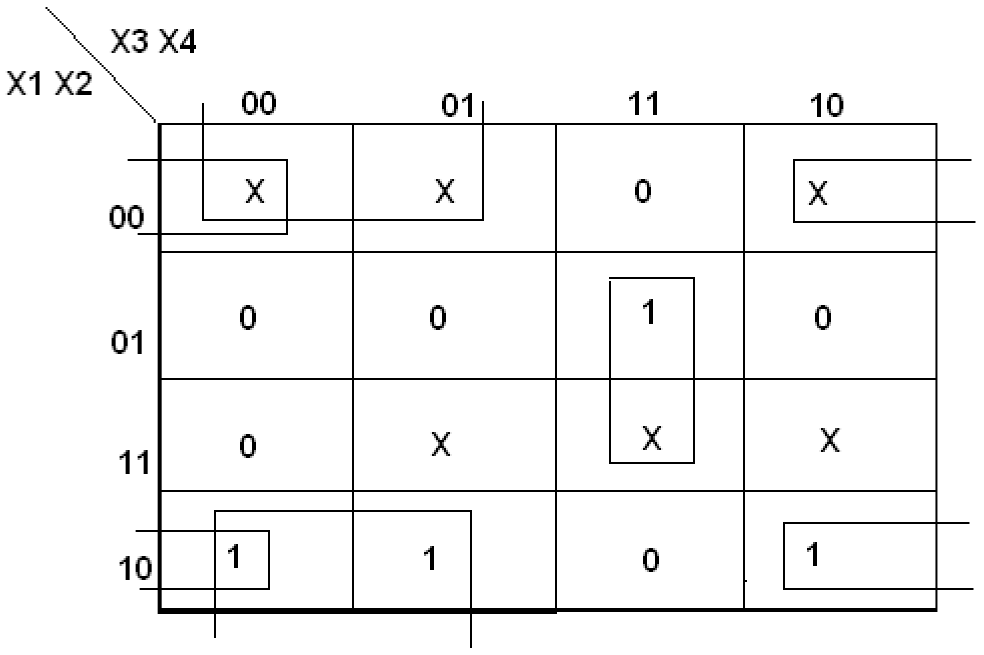
Excess – 3 Input				BCD Output			
B3	**B2**	**B1**	**B0**	**G3**	**G2**	**G1**	**G0**
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1

K-MAP FOR A



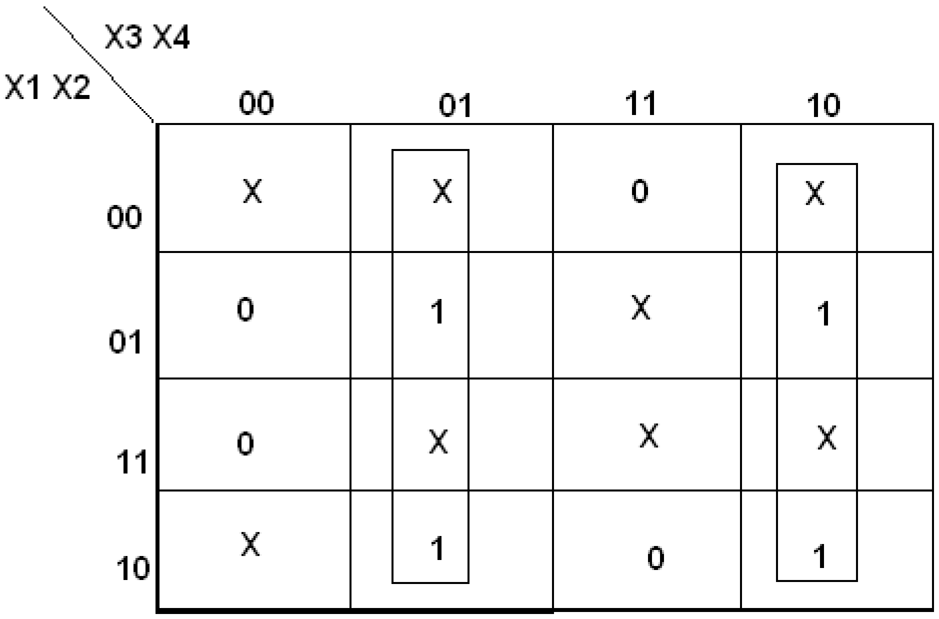
A = X1 X2 + X3 X4 X1

K-MAP FOR B



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

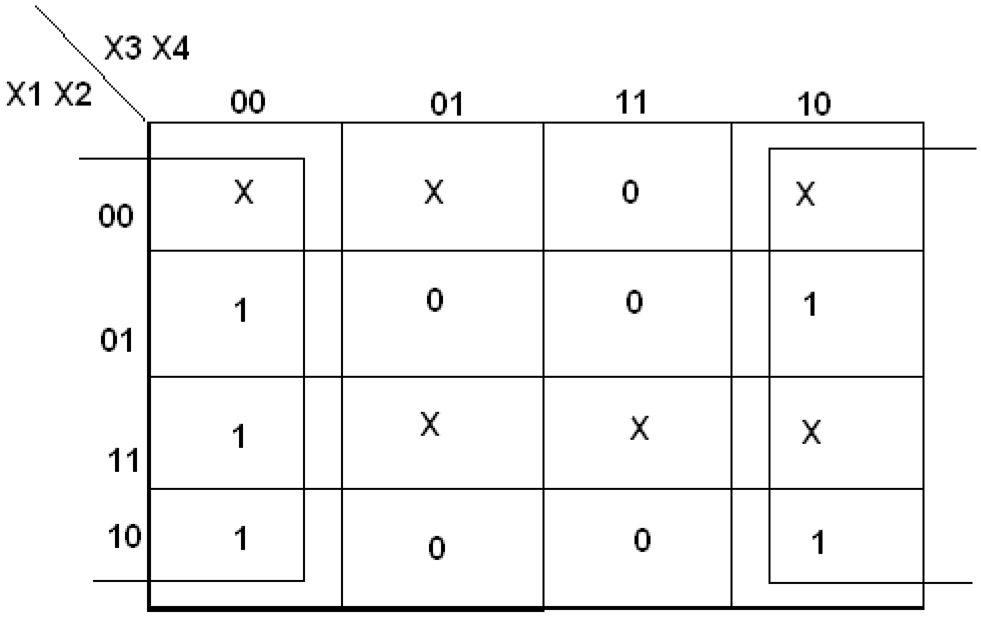
K-MAP FOR C



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

$$C = X3 \oplus X4$$

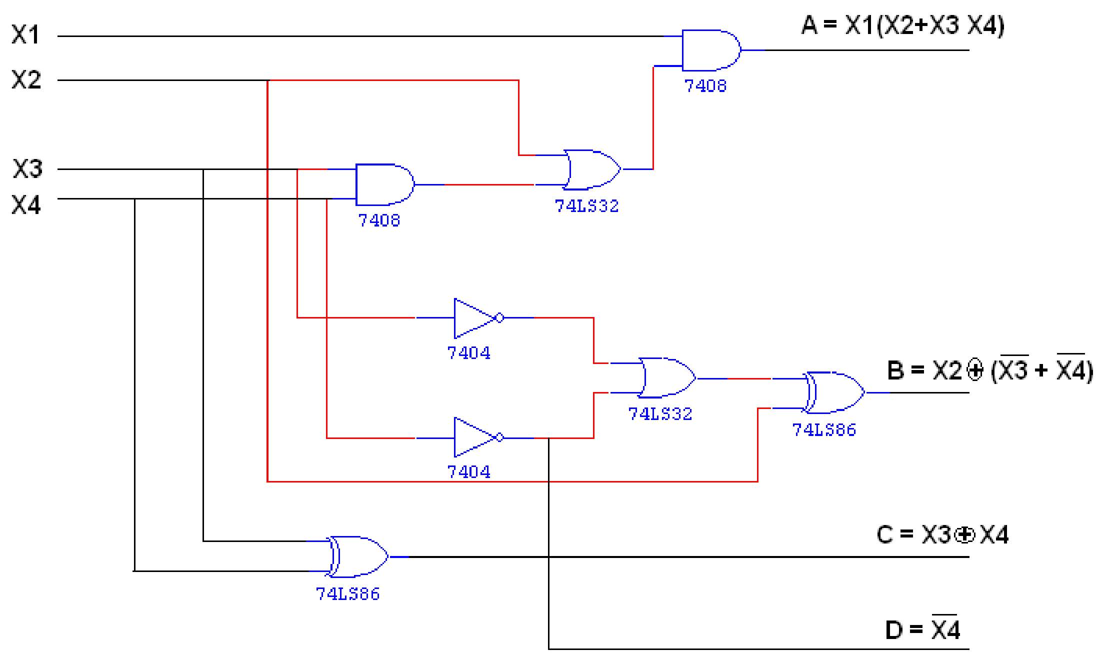
K-MAP FOR D



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

$$D = \overline{X4}$$

LOGIC DIAGRAM



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

Thus, the code converter are studied and verified using logic gates

DESIGN OF 4-BIT ADDER AND SUBTRACTOR

EXP NO. : 5
DATE :

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	-	As per Requirement

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.

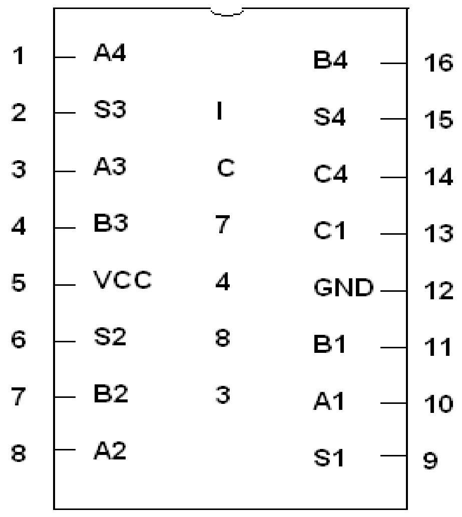
4 BIT BINARY SUBTRACTOR

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.

PIN DIAGRAM FOR IC 7483

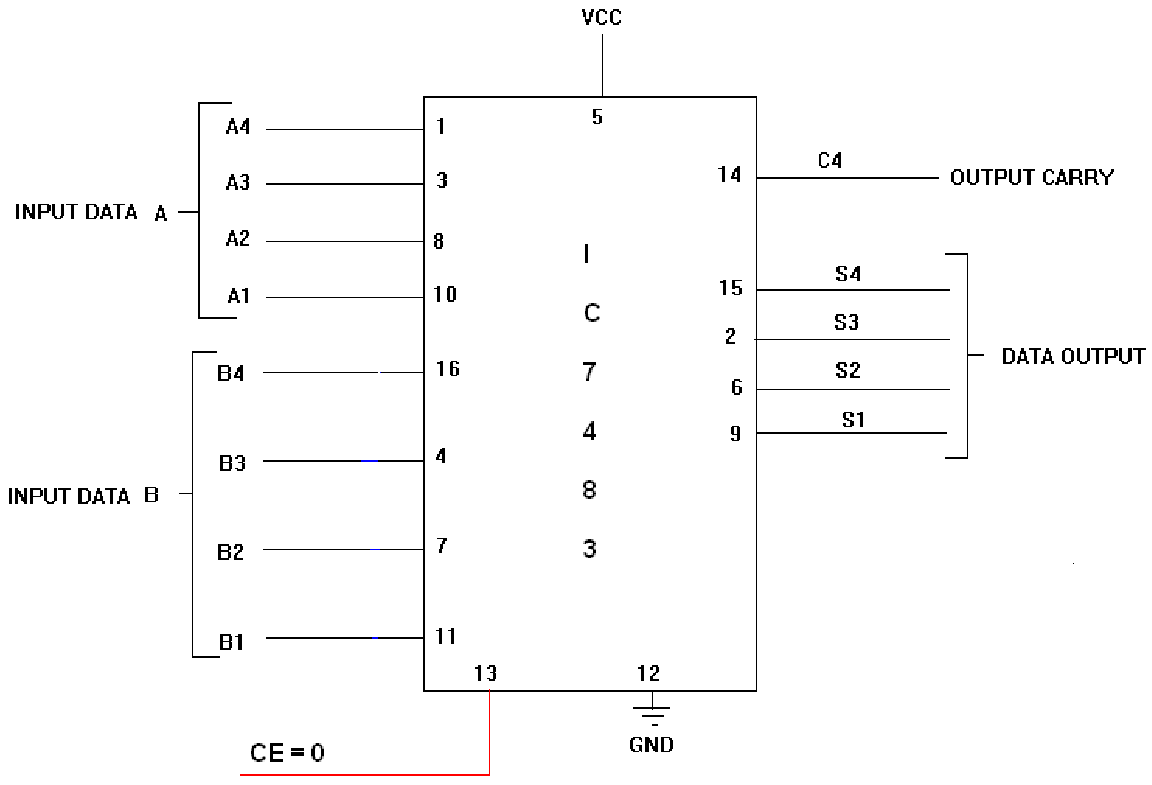


TRUTH TABLE

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

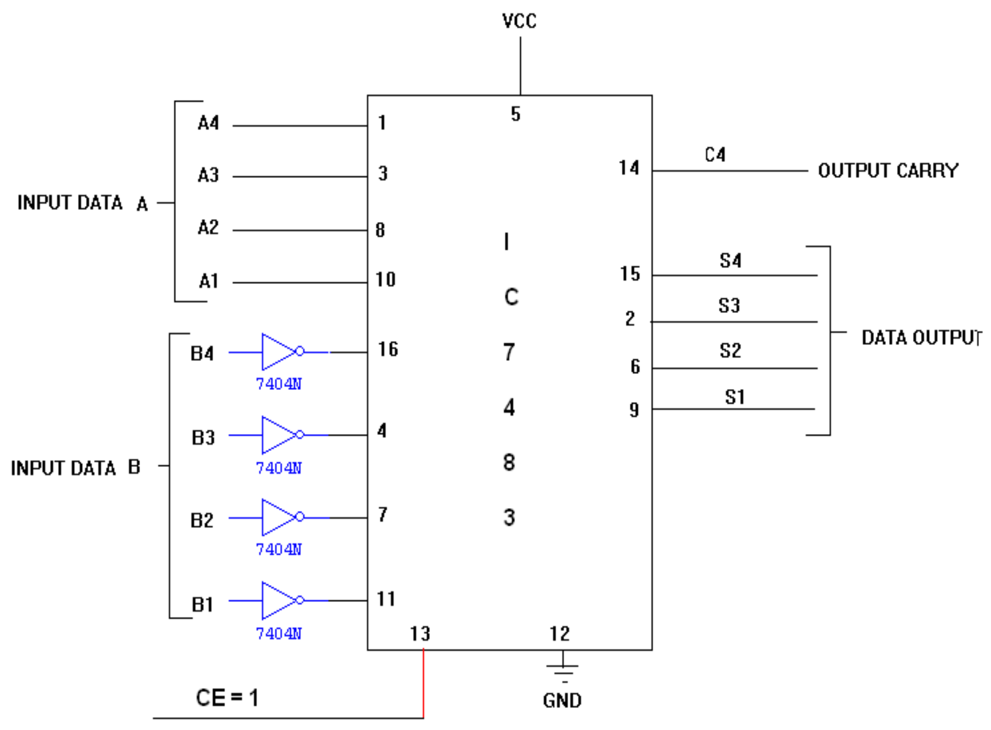
LOGIC DIAGRAM

4-BIT BINARY ADDER



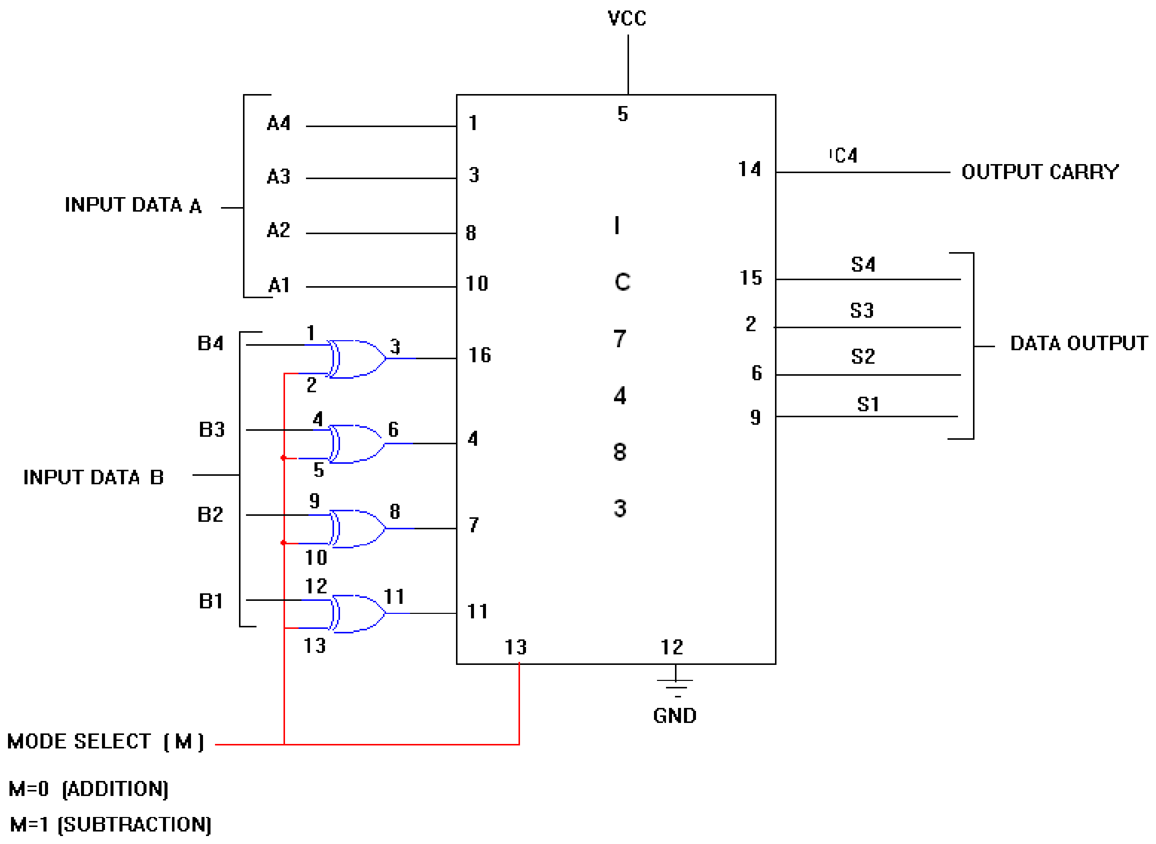
LOGIC DIAGRAM

4-BIT BINARY SUBTRACTOR



LOGIC DIAGRAM

4-BIT BINARY ADDER/SUBTRACTOR



PROCEDURE

(iv) Connections were given as per circuit diagram.

(v) Logical inputs were given as per truth table

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

RESULT

Thus, 4 bit Adder and Subtractor are designed and verified using logic gates

DESIGN OF BCD ADDER

EXP NO. : 6

DATE :

AIM

To design and implement BCD adder using IC 7483.

APPARATUS REQUIRED

SL.NO.	COMPONENT	SPECIFICATION	QTY.
1.	IC	IC 7483	2
2.	OR GATE	IC 7432	1
3.	AND GATE	IC 7408	1
3.	IC TRAINER KIT	-	1
4.	PATCH CORDS	-	As per Requirement

THEORY:

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.

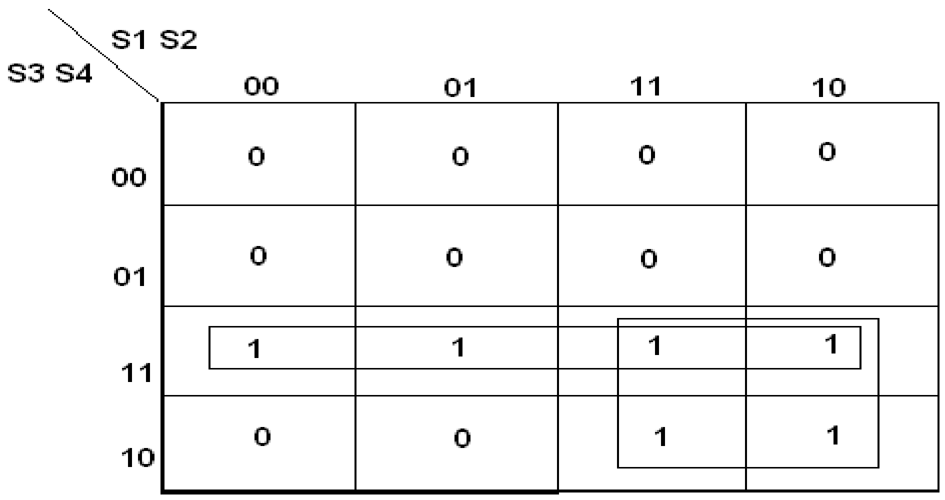
TRUTH TABLE

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

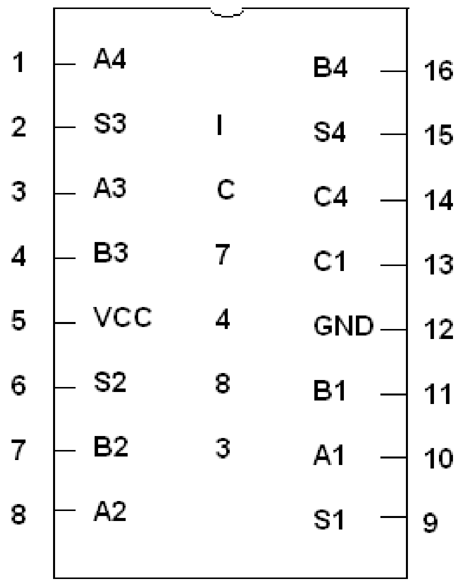
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

K MAP

Y = S3(S4 + S1)

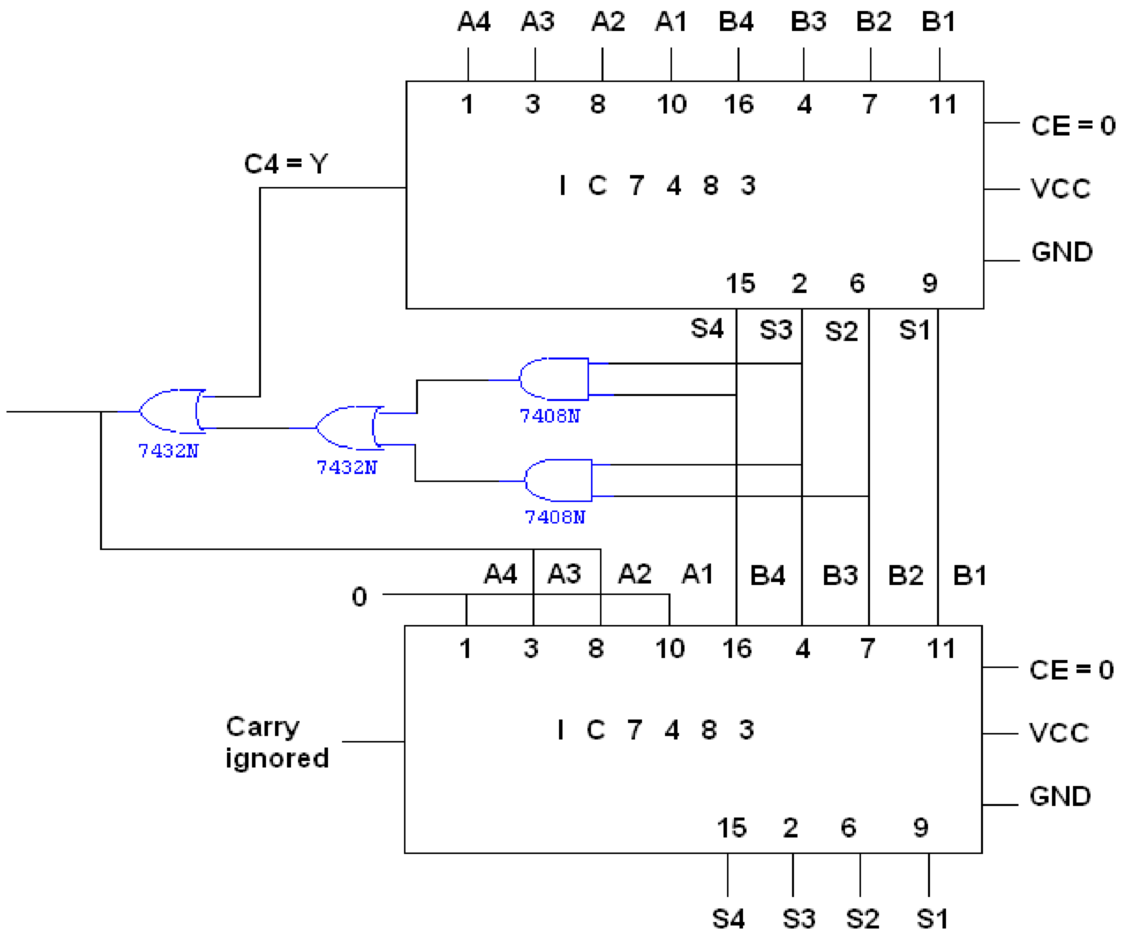


PIN DIAGRAM FOR IC 7483



LOGIC DIAGRAM

BCD ADDER



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:

Thus, BCD Adder is designed and verified using logic gates

DESIGN AND IMPLEMENTATION OF MAGNITUDE COMPARATOR

EXP NO. : 7

DATE :

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	-	As per Requirement

THEORY

The comparison of two numbers is operators 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 determines their relative magnitude. The outcome of the comparator is specified by three binary variables that indicate whether A>B, A=B (or) A<B.

$A = A_3 \ A_2 \ A_1 \ A_0$

$B = B_3 \ B_2 \ B_1 \ B_0$

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

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

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

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

Where, A = B is expanded as,

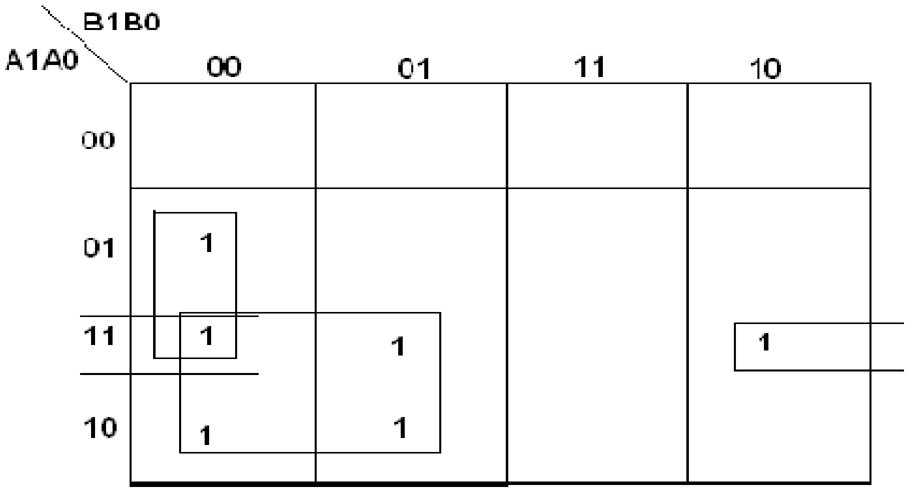
$$\begin{tikzcd}
A = B = (A_3 + B_3) \arrow[d] & (A_2 + B_2) \arrow[d] & (A_1 + B_1) \arrow[d] & (A_0 + B_0) \arrow[d] \\
x_3 & x_2 & x_1 & x_0
\end{tikzcd}$$

2 BIT MAGNITUDE COMPARATOR

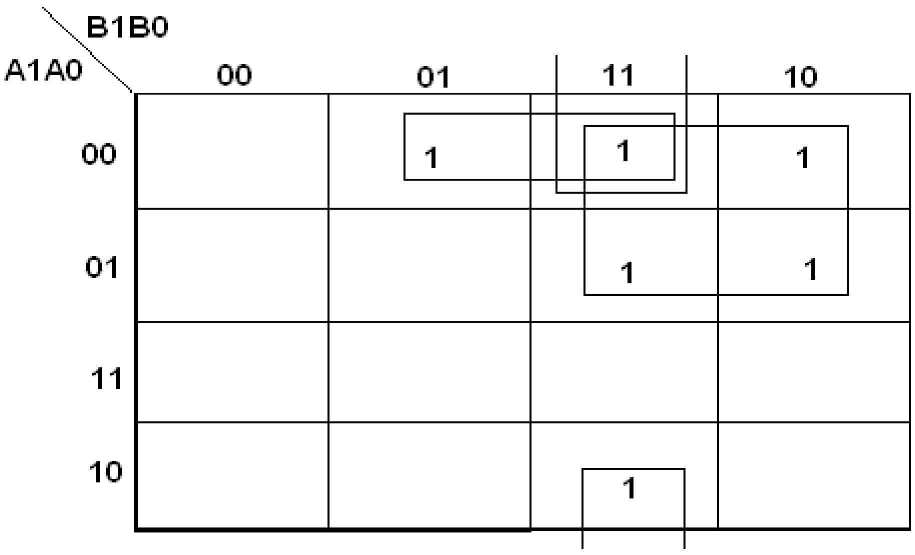
TRUTH TABLE

A1	A0	B1	B0	A > B	A = B	A < B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

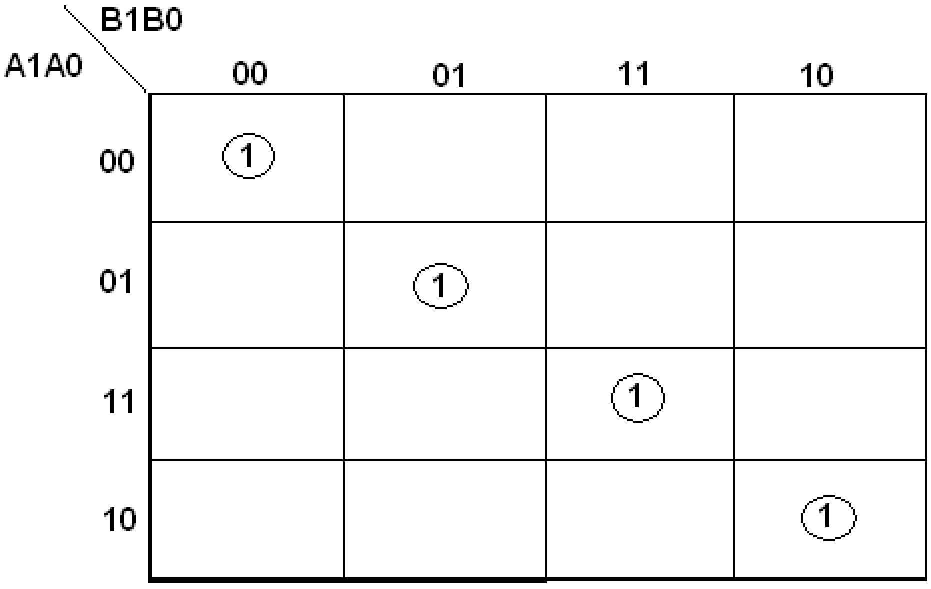
K MAP



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

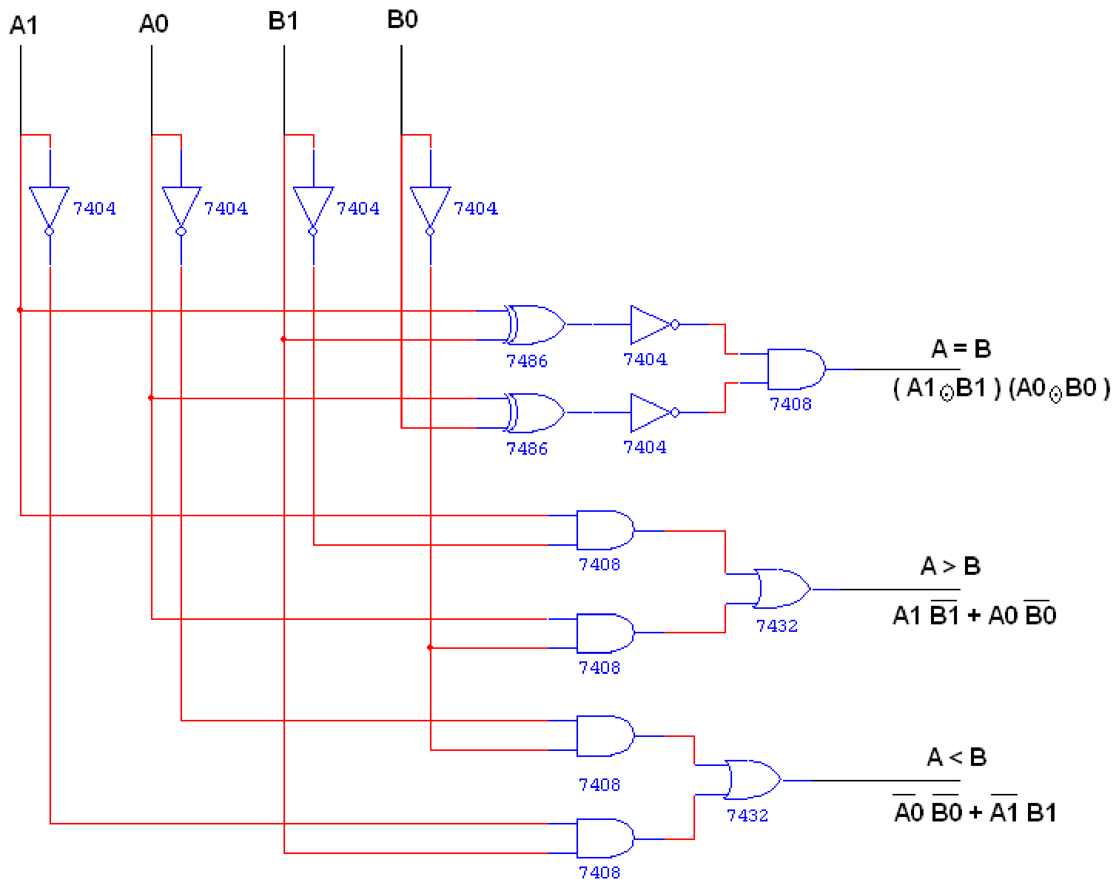


$A < B = \overline{A1}\;\overline{A0}\;B0 + \overline{A0}\;B0\;B1 + \overline{A1}\;B1$

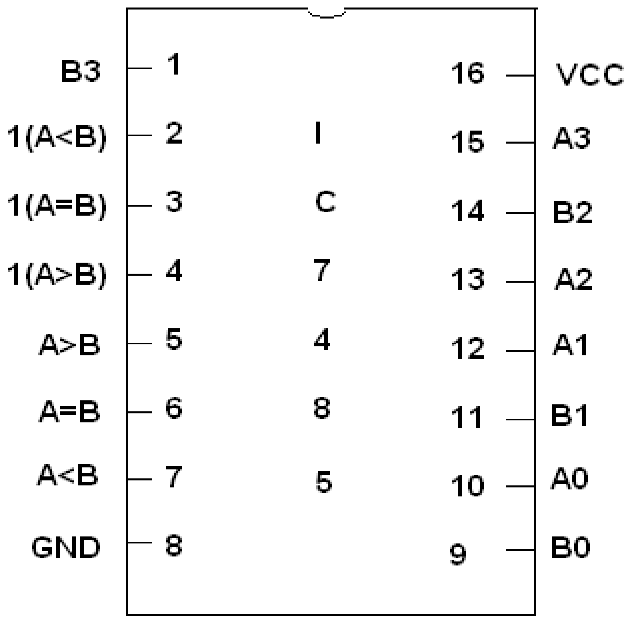


$A = B = (A0 \odot B0)(A1 \odot B1)$

LOGIC DIAGRAM



PIN DIAGRAM FOR IC 7485

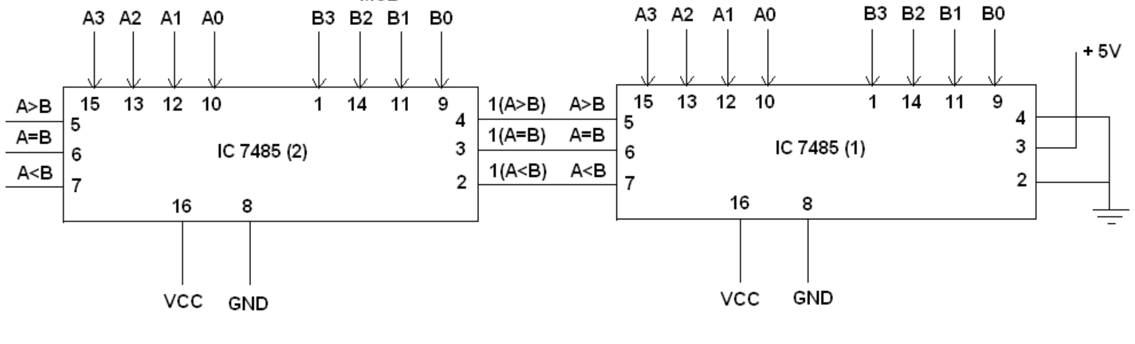


8 BIT MAGNITUDE COMPARATOR

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

LOGIC DIAGRAM



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

Thus, magnitude comparator are designed and verified using logic gates

DESIGN AND IMPLEMENTATION OF MULTIPLEXER AND DEMULTIPLEXER

EXP NO. : 8

DATE :

AIM

To design and implement multiplexer and de multiplexer using logic gates.

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	-	As per Requirement

THEORY

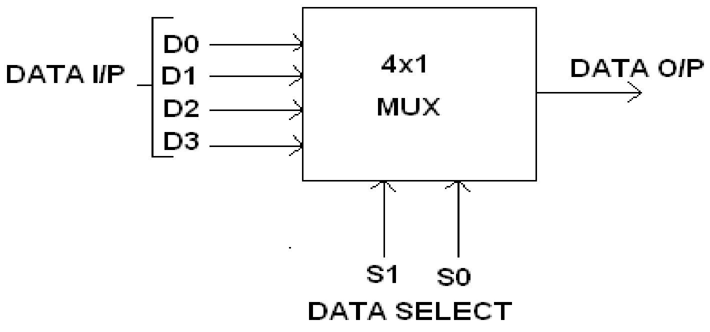
MULTIPLEXER

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 de multiplexer. 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

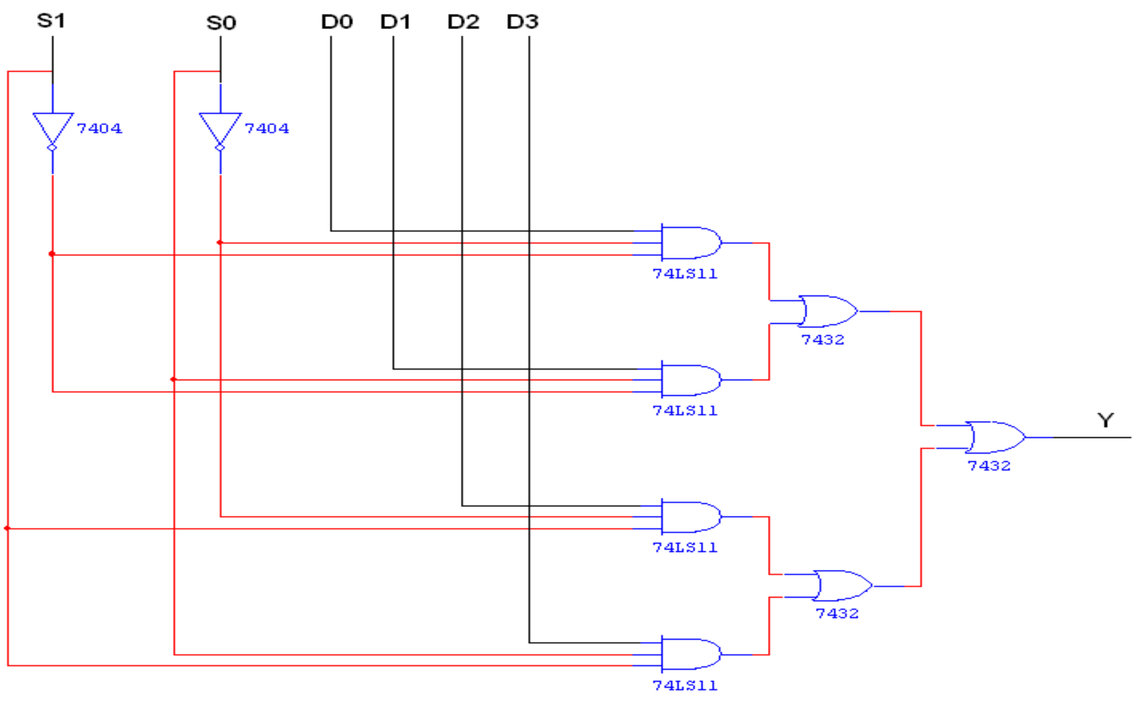
S1	S0	INPUTS Y
0	0	D0 → D0 S1’ S0’
0	1	D1 → D1 S1’ S0
1	0	D2 → D2 S1 S0’
1	1	D3 → D3 S1 S0

Y = D0 S1’ S0’ + D1 S1’ S0 + D2 S1 S0’ + D3 S1 S0

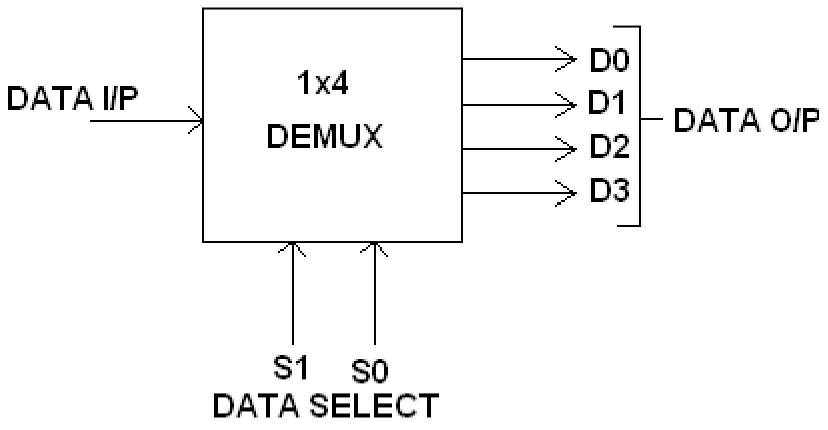
TRUTH TABLE

S1	S0	Y = OUTPUT
0	0	D0
0	1	D1
1	0	D2
1	1	D3

CIRCUIT DIAGRAM FOR MULTIPLEXER



BLOCK DIAGRAM FOR 1:4 DE MULTIPLEXER



FUNCTION TABLE

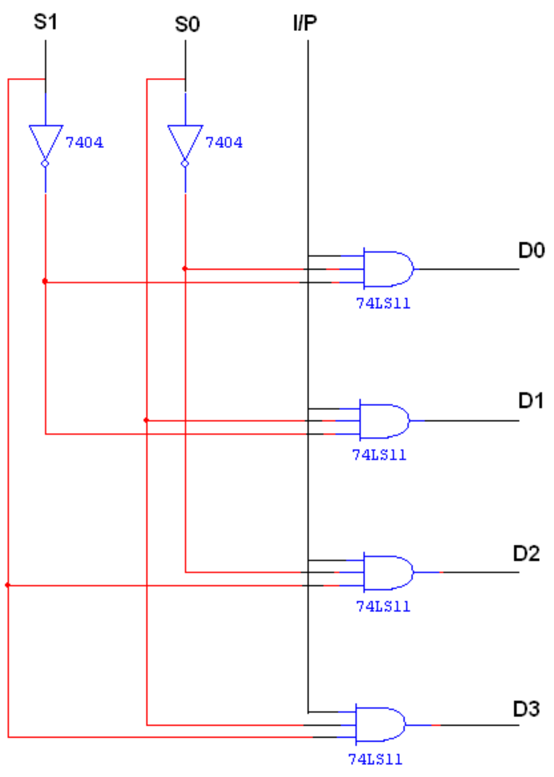
S1	S0	INPUT
0	0	X → D0 = X S1’ S0’
0	1	X → D1 = X S1’ S0
1	0	X → D2 = X S1 S0’
1	1	X → D3 = X S1 S0

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

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

LOGIC DIAGRAM FOR DEMULTIPLEXER



PROCEDURE

(iv) Connections are given as per circuit diagram.

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

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

RESULT :

Thus, Multiplexer and Demultiplexer are designed and verified using logic gates

DESIGN AND IMPLEMENTATION OF ENCODER,PRIORITY ENCODER AND DECODER

EXP NO. : 9

DATE :

AIM

To design and implement encoder ,Priority encoder and decoder using logic gates.

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	-	As per Requirement

THEORY

ENCODER

An encoder is a digital circuit that performs 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 $2^n – 1$.

PRIORITY ENCODER:

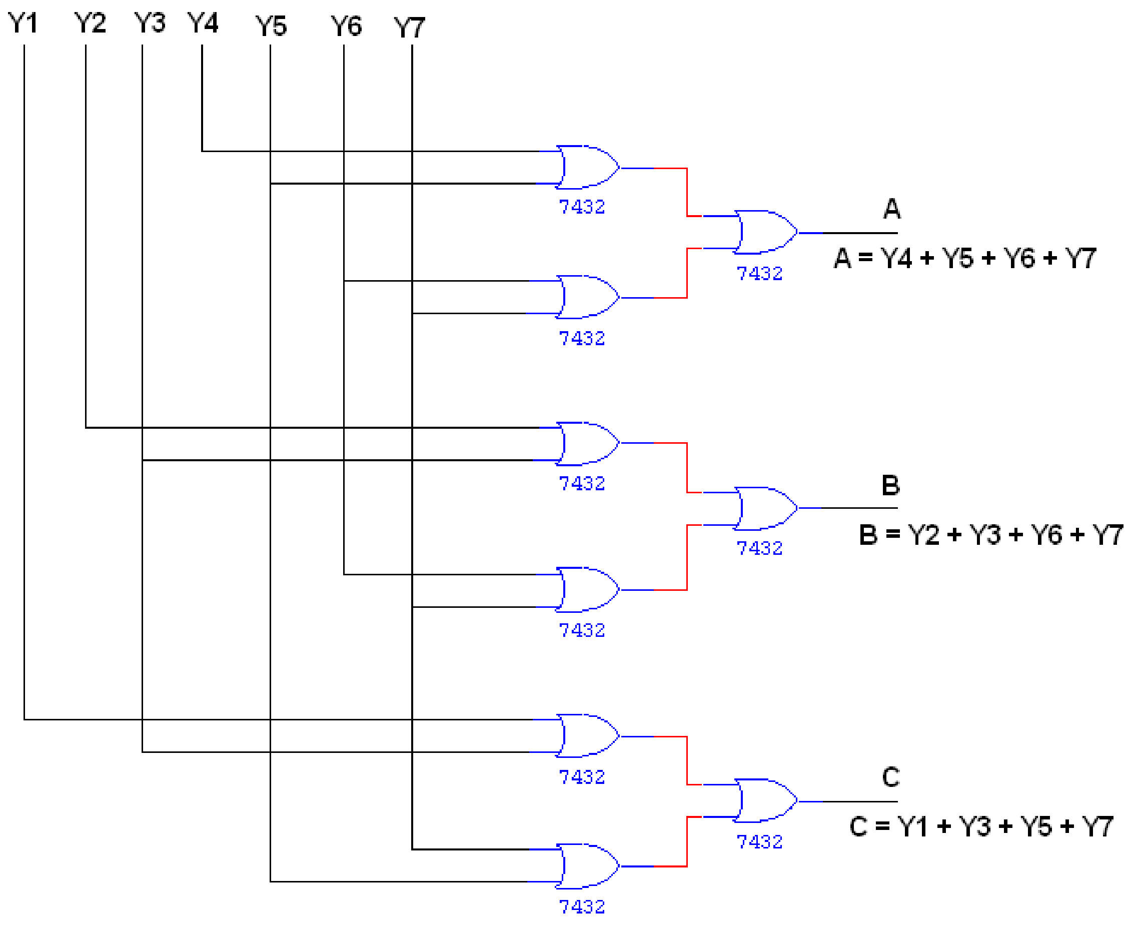
A **priority encoder** is a circuit or algorithm that compresses multiple binary inputs into a smaller number of outputs. The output of a **priority encoder** is the binary representation of the original number starting from zero of the most significant input bit.

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 ENCODER

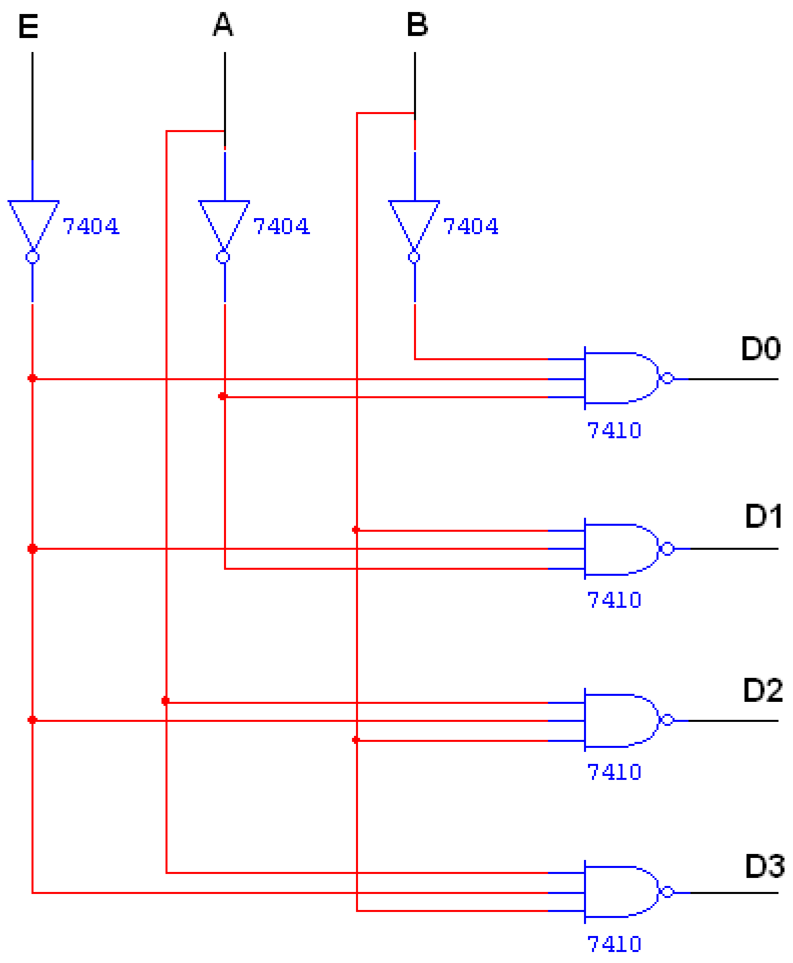


DECODER:

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

LOGIC DIAGRAM FOR DECODER

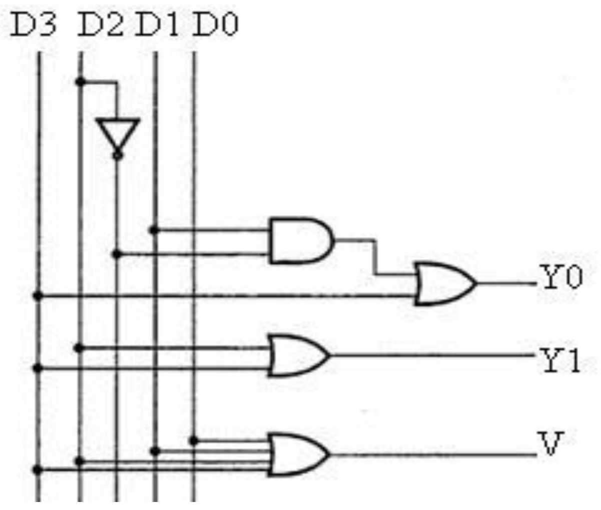


PRIORITY ENCODER:

TRUTH TABLE

Input				Output		
D0	**D1**	**D2**	**D3**	**Y1**	**Y0**	**V**
0	0	0	0	0	0	0
1	0	0	0	1	0	1
X	1	0	1	0	1	1
X	X	1	1	1	1	1
X	X	X	1	1	1	1

LOGIC DIAGRAM FOR PRIORITY ENCODER



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:

Thus, Encoder, Decoder and Priority Encoder are designed and verified using logic gates

VERIFICATION OF FLIP FLOPS

EXP NO. : 10
DATE :

AIM:

To Verify and study the working of the following flip-flops:

- **RS flip-flop**
- **D flip-flop**
- **JK flip-flop**
- **T flip-flop**

APPARATUS REQUIRED

S.No.	Apparatus	Specifications	Quantity

1.	**IC Trainer kit**	**--**	**1 no**
2.	**Logic gate IC's**	**IC 7408, IC 7402, IC 7400**	**1no each**
3.	**Connecting wires**	**--**	**1 set**

THEORY:

A Flip-Flop is a bistable device, with inputs, that remains in a given state as long as power is applied and until input signals are applied to cause its output to change. They are memory devices that are capable of storing logic constants. The process of storing a 1 into a flip-flop is called setting or presetting the flip-flop; while the process of storing a 0 into the flip-flop is called resetting or clearing the flip-flop. The inputs to the flip-flops are of two types:

- Asynchronous or direct inputs – input signal change produces an immediate change in the state of the flip-flop
- Synchronous inputs – input signal change does not affect the state of the flip-flop immediately, but rather affects the state of the flip-flop only when some control signal, usually called an enable or clock input also occurs.

Clocked flip-flops are designed to cause an output change either when the clock signal makes a transition or when the signal reaches a particular level. Accordingly it is called as an Edge triggered or Level triggered flip-flop.

A special class of flip-flops called as Latches are characterized by the fact that the timing of the output changes are not controlled i.e., the output essentially responds immediately to changes on the input lines, although a special control signal, called enable or clock, might also need to be present.

RS Flip-Flop:

This form of flip-flop has two input lines and one or more output lines. The output line that is always present is labeled as Q. Generally, a second output called as Q' will also be present. One input line S=1, R=0 is used to set the device to Q=1 state, While the other input R=1, S=0 is used to reset the device to Q=0 state. The Q and Q' make up what is called as *double-rail output*, that is Q=Q', this condition is normally avoided in the actual circuit operation.

D Flip-Flop:

In some applications, the S and R inputs will always be complementary, i.e., R=0 when S=1 and R=1 when S=0, this can be expressed as S=R'. Because pin connection can be minimized on an IC chip for this circuit, this circuit has become a popular device known as D flip-flop. The D flip-flop is called as a *Transparent flip-flop* because the Q output follows the D input i.e., Q=1 when D=1 & Q=0 when D=0.

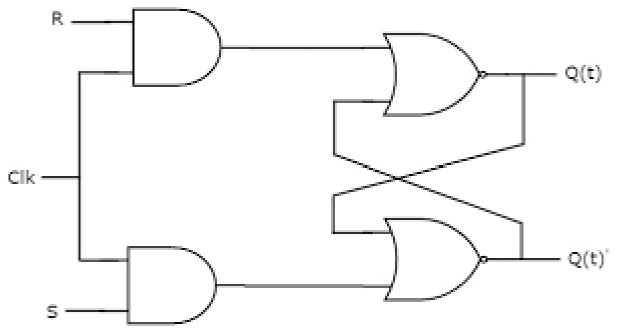
JK Flip-Flop:

This bistable circuit has two gating inputs along with a clock input. The voltage level of the gates determines the output state to which the clock input will shift the flip-flop. The Set condition of flip-flop is achieved when J=1,K=0 and the Reset condition achieved when J=0,K=1. When both J=K=0 the condition is called as *'no change'.* When both J=K=1 the clock input causes the flip-flop to *'toggle'* (complement of the previous state). When both J=K=1 and the clock pulse is activated for quite a long time then the output of the flip-flop keeps toggling which is a *Race Condition.* This can be avoided by choosing the clock pulse interval to be lesser than the propagation delay of the flip-flop.

T Flip-Flop:

A multivibrator that changes state or toggles with each successive input is called a T or Toggle flip-flop. If T=1, positive or negative transitions of the clock input will cause the output to change state. If T=0, then clock input has no effect on the flip-flop output. Manufacturers do to produce T flip-flops. Instead, they are created from JK flip-flop by tying together J and K inputs.

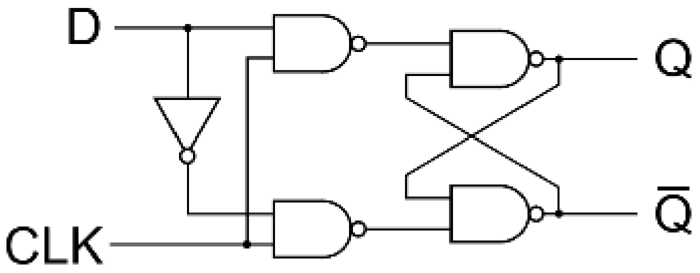
RS FLIP - FLOP



TRUTH TABLE

R	S	Q_{n+1}
0	0	Q_n
0	1	1
1	0	0
1	1	X

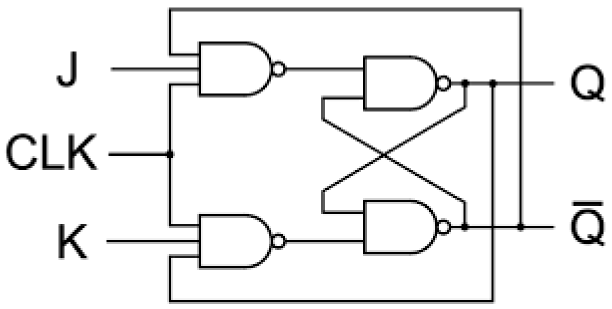
D FLIP – FLOP



TRUTH TABLE

D	Q_{n+1}
0	0
1	1

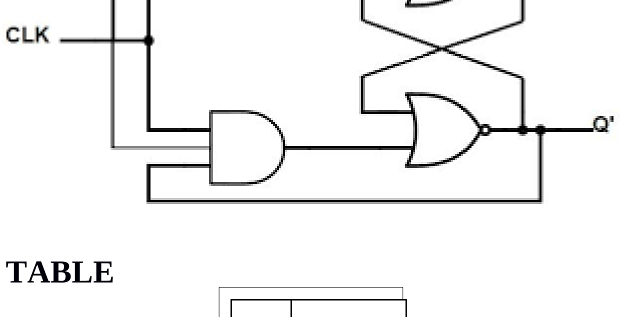
JK FLIP – FLOP



TRUTH TABLE

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	Qn^1

T FLIP – FLOP



TRUTH TABLE

T	Q_{n+1}
0	Q_n
1	Qn^1

PROCEDURE:

1. The connections are made as per the circuit diagram
2. The clock input is given to the clocked flip-flops
3. The inputs are varied and the truth table of each flip-flop is verified using the output indications

RESULT:

Thus the flip-flops were constructed and their truth tables were verified.

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

EXP NO. : 11

DATE :

AIM

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

APPARATUS REQUIRED

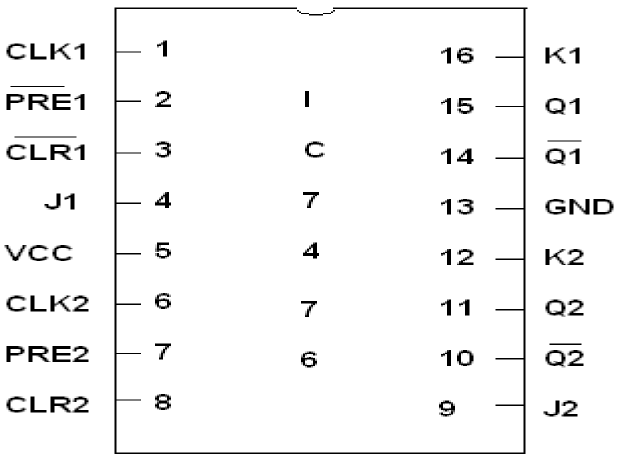
SL.NO.	COMPONENT	SPECIFICATION	QTY.

1.	JK FLIP FLOP	IC 7476	2
2.	NAND GATE	IC 7400	1
3.	IC TRAINER KIT	-	1
4.	PATCH CORDS	-	As per Requirement

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 $\overline{Q}$ 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



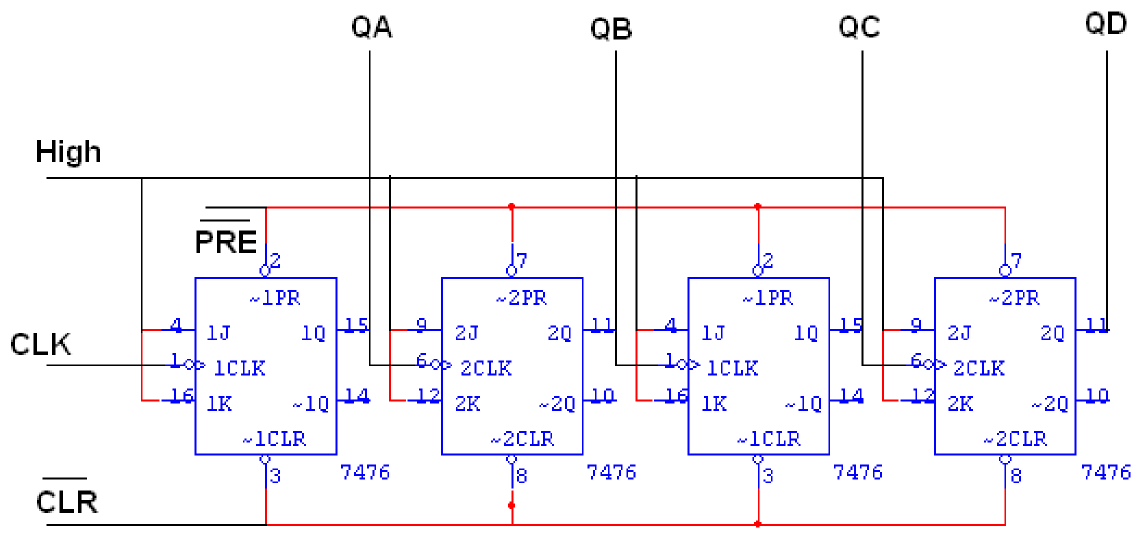
4 BIT RIPPLE COUNTER:

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 4 BIT RIPPLE COUNTER:

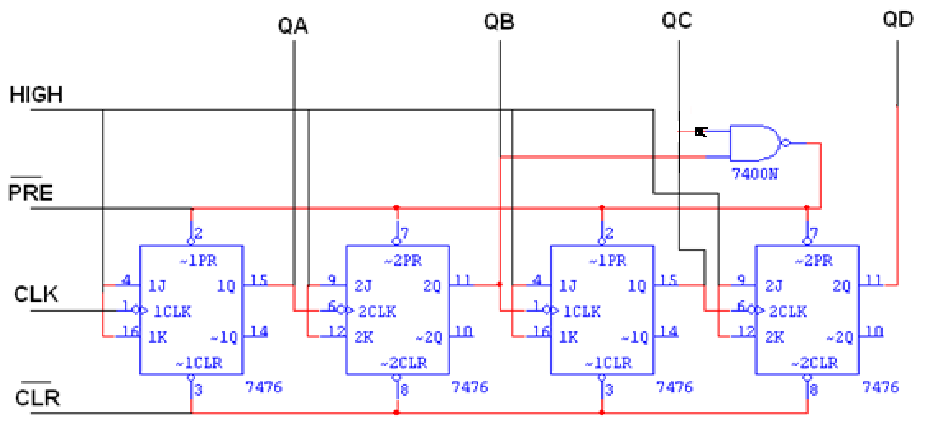


MOD - 10 RIPPLE COUNTER:

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 - 10 RIPPLE COUNTER

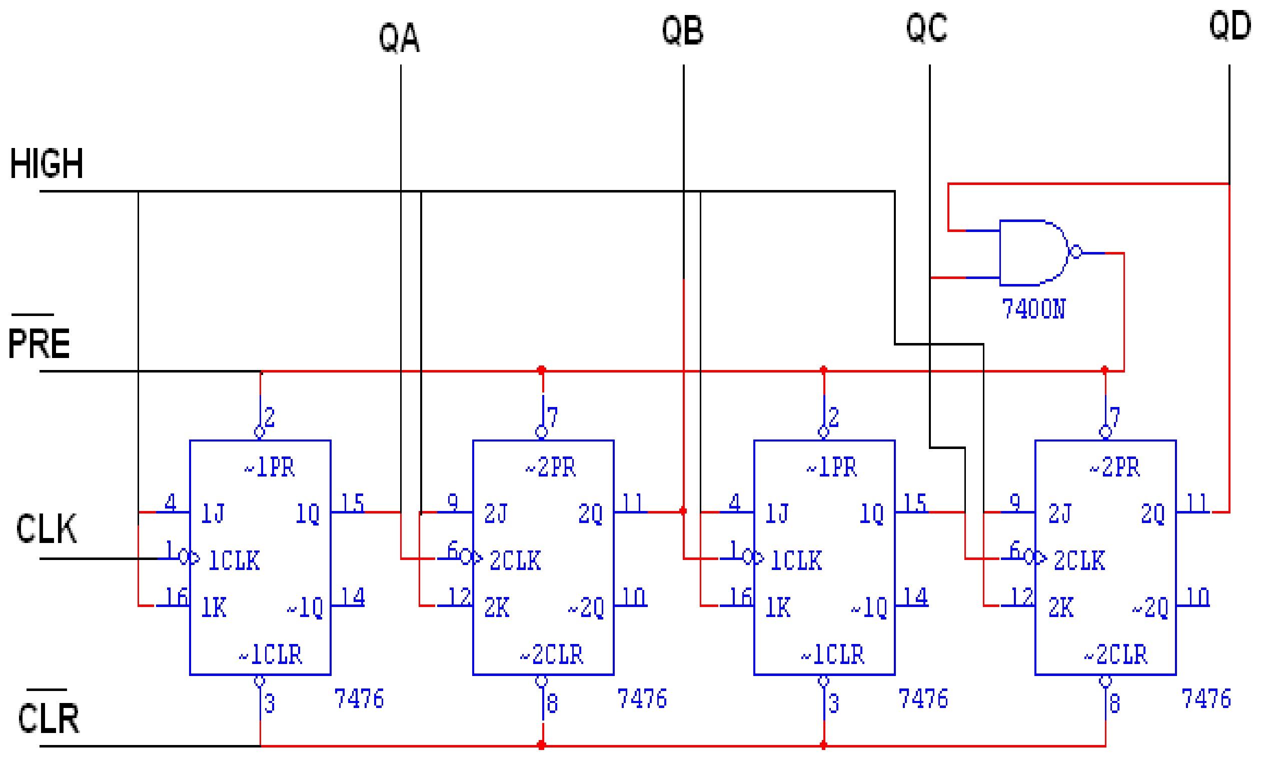


MOD - 12 RIPPLE COUNTER:

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

LOGIC DIAGRAM FOR MOD - 12 RIPPLE COUNTER



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 :

Thus the 4 bit Ripple counter and Mod 10/12 counter were constructed and their truth tables were verified

DESIGN AND IMPLEMENTATION OF 4-BIT RING COUNTER

EXP NO. : 12

DATE :

AIM:

To design and verify 4-bit ring counter using IC 7474.

APPARATUS REQUIRED

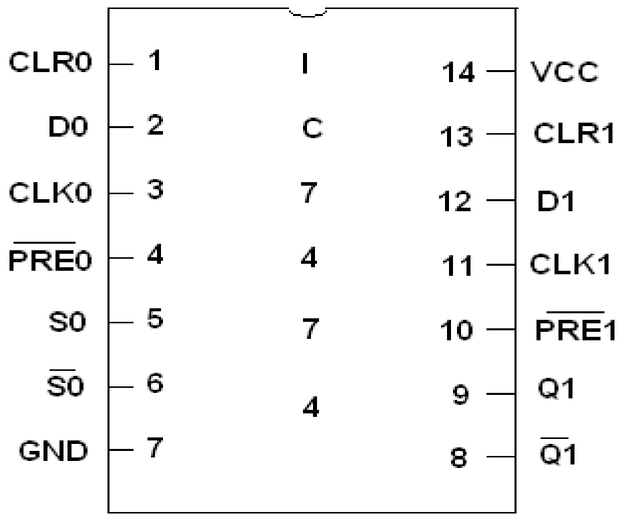
SL.NO.	COMPONENT	SPECIFICATION	QTY.
1.	D FLIP FLOP	IC 7474	2
2.	IC TRAINER KIT	-	1
3.	PATCH CORDS	-	As per Requirement

THEORY:

The ring counter is a cascaded connection of flip flops, in which the output of last flip flop is connected to input of first flip flop. In ring counter if the output of any stage is 1, then its reminder is 0. The Ring counters transfers the same output throughout the circuit.

That means if the output of the first flip flop is 1, then this is transferred to its next stage i.e. 2nd flip flop. By transferring the output to its next stage, the output of first flip flop becomes 0. And this process continues for all the stages of a ring counter. If we use n flip flops in the ring counter, the '1' is circulated for every n clock cycles.

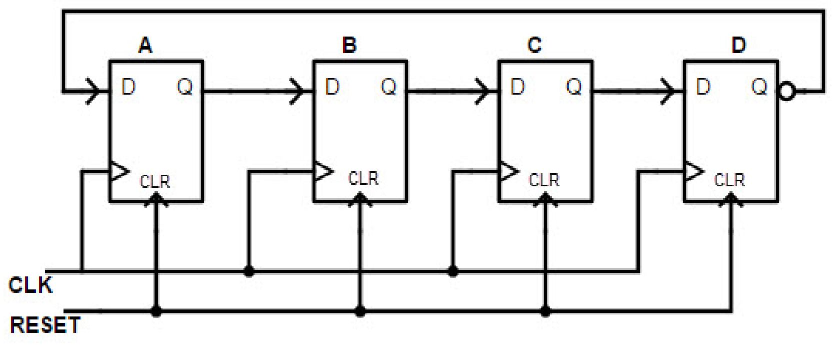
PIN DIAGRAM OF IC 7474:



TRUTH TABLE:

CLK	QA	**QB**	**QC**	**QD**
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	0	0	0

LOGIC DIAGRAM FOR 4-BIT RING COUNTER:



PROCEDURE

(iv) Connections are given as per circuit diagram.

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

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

RESULT :

Thus the 4 bit Ring counter were constructed and their truth tables were verified

CODING OF COMBINATIONAL CIRCUITS USING HDL

EXP NO. : 13

DATE :

AIM:

To write Verilog programs for combinational circuits.

SOFTWARE REQUIRED:

1. Xilinx ISE 10.1

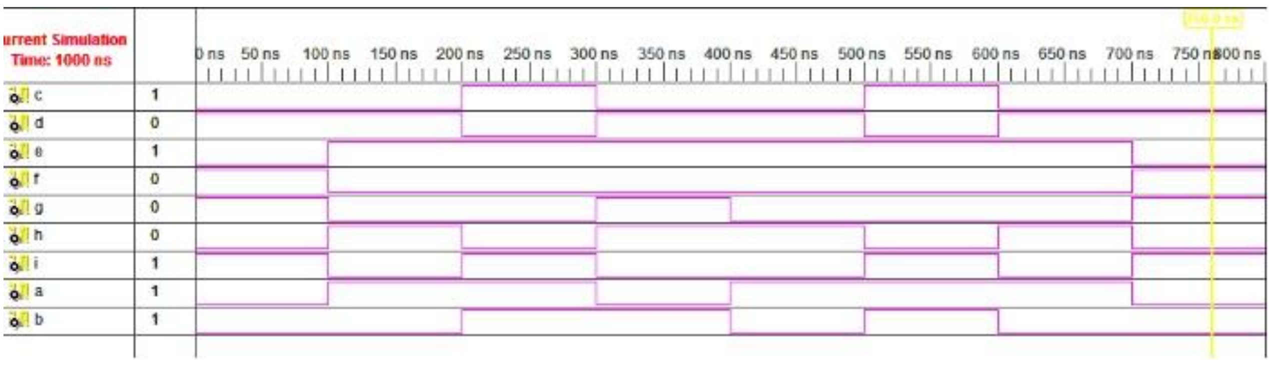
2. Simulation Tools

PROGRAM:

LOGIC GATES:

```
module logicgates(a,b,c,d,e,f,g,h,i);
input a,b;
output c,d,e,f,g,h,i;
and(c,a,b);
or(d,a,b);
xor(e,a,b);
nand(f,a,b);
nor(g,a,b);
xnor(h,a,b);
not(i,a);
endmodule
```

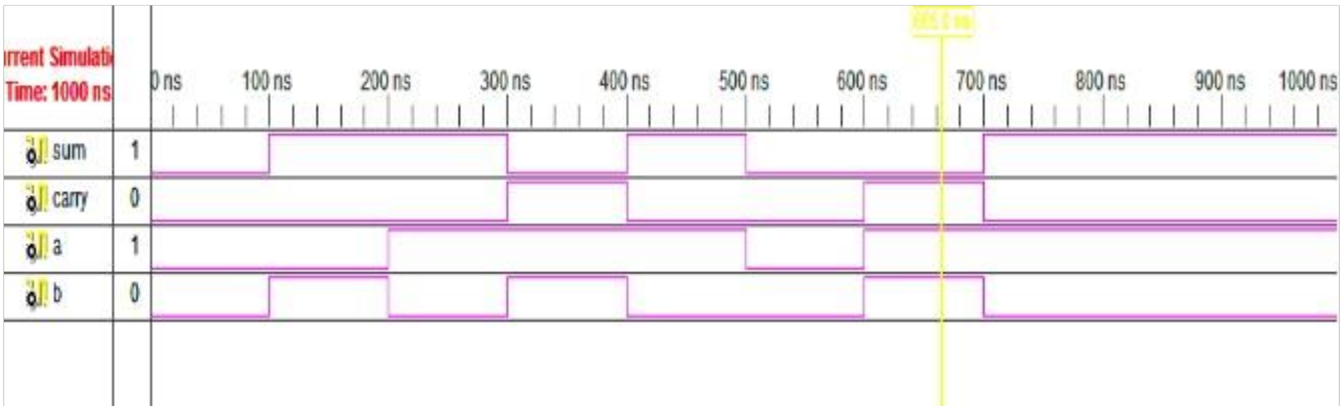
OUTPUT:



HALF ADDER

```
module ha(a,b,sum,carry);
input a,b;
output sum,carry;
xor (sum,a,b);
and (carry,a,b);
endmodule
```

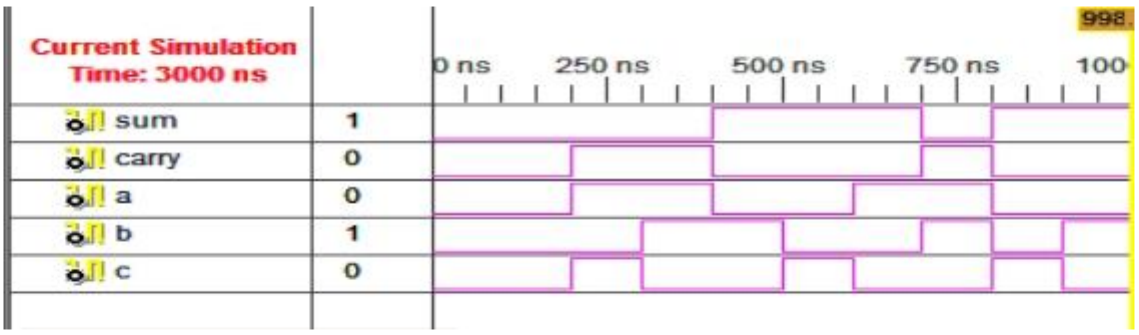
OUTPUT:



FULL ADDER

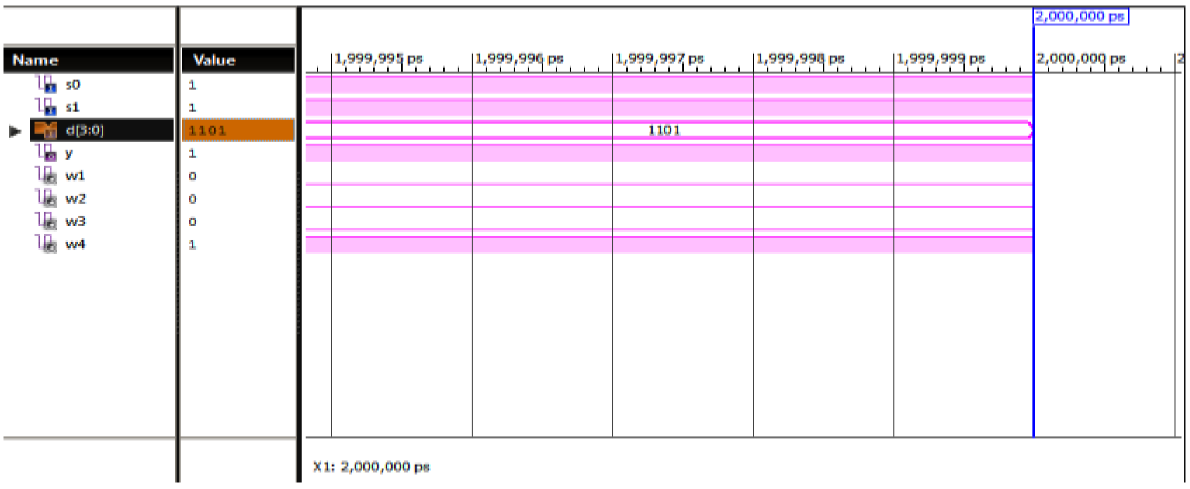
```
module fa(a,b,c,sum,carry);
input a,b,c;
output sum,carry;
wire [3:1]w;
xor g1(w[1],a,b);
xor g3(sum,w[1],c);
and g2(w[2],a,b);
and g4(w[3],w[1],c);
or g5(carry,w[3],w[2]);
endmodule
```

OUTPUT:



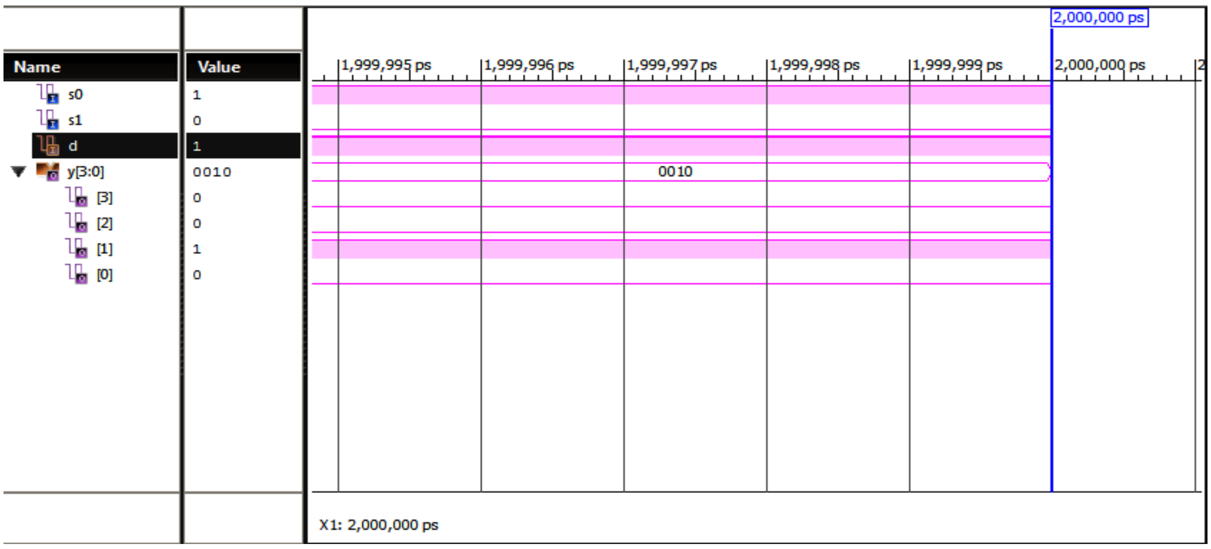
MULTIPLEXER:

```
module mux_4(y,s0,s1,d);
input s0,s1;
input [3:0] d;
output y;
wire w1,w2,w3,w4;
and g1(w1,(~s1),(~s0),d[0]);
and  g2(w2,(~s1),(s0),d[1]);
and  g3(w3,(s1),(~s0),d[2]);
and g4(w4,(s1),(s0),d[3]);
or g5(y,w1,w2,w3,w4);
endmodle
```



DEMULTIPLEXER:

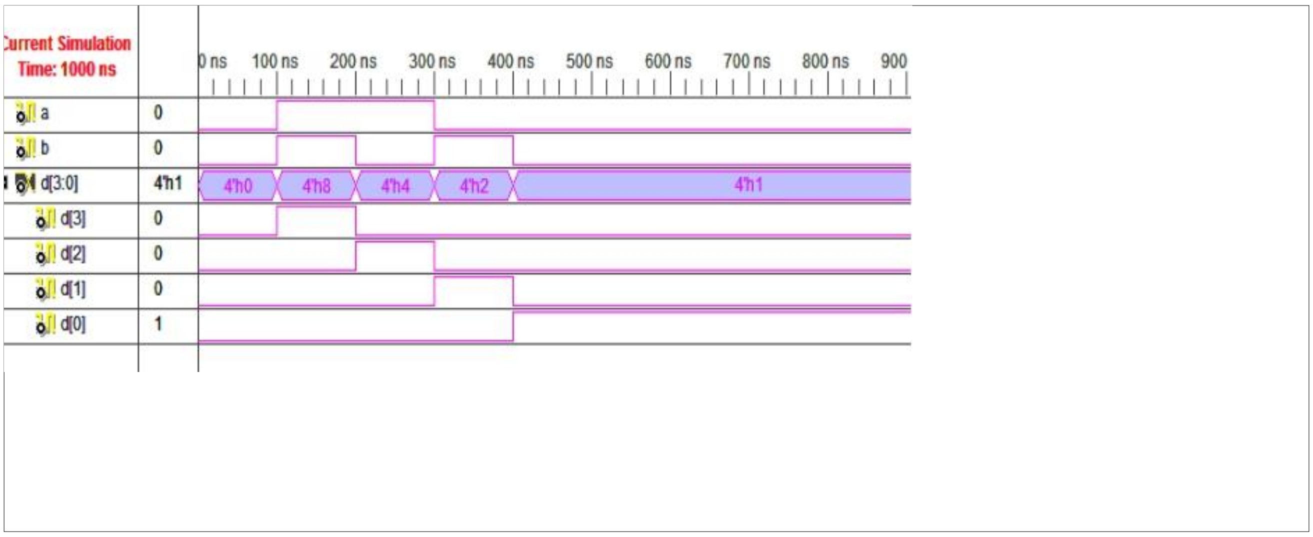
```
module
demux_4(y,s0,s1,d);
input s0,s1,d;
output [3:0]y;
and g1(y[0],(~s1),(~s0),d);
and g2(y[1],(~s1),(s0),d);
and g3(y[2],(s1),(~s0),d);
and g4(y[3],(s1),(s0),d);
endmodule
```



ENCODER

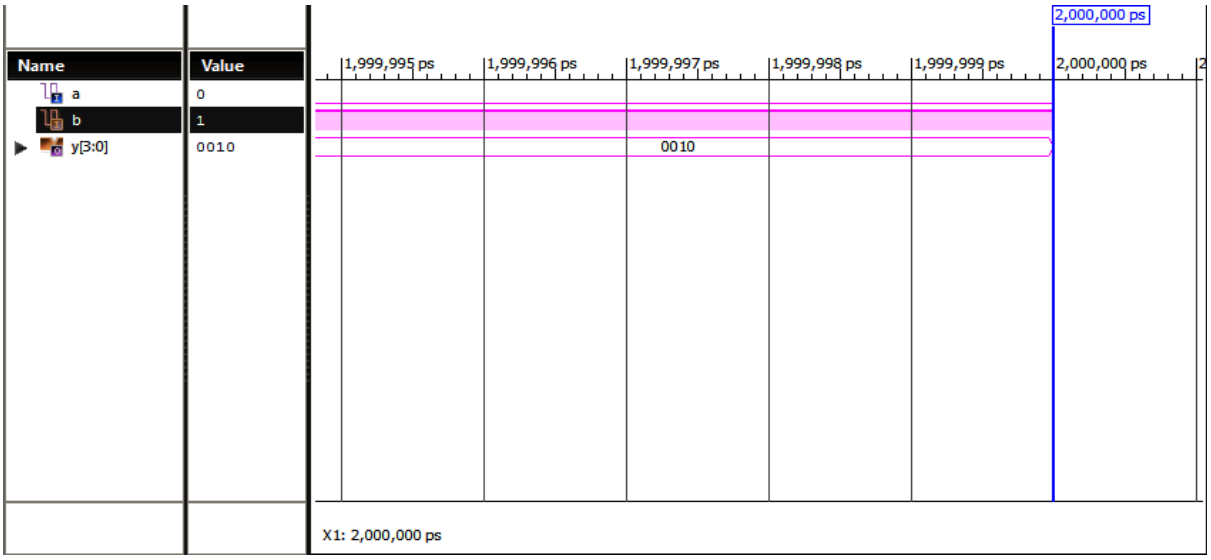
```
module encoder_4(d,a,b);
input [3:0]d;
output a,b;
or g1(a,d[2],d[3]);
or g2(b,d[1],d[3]);
endmodule
```

OUTPUT:



DECODER:

```
module decoder_4(a,b,y);
input a,b;
output [3:0]y;
and g1(y[0],(~a),(~b));
and g2(y[1],(~a),(b));
and g3(y[2],(a),(~b));
and g4(y[3],(a),(b));
endmodule
```



RESULT:

Thus the simulation of combinational circuits was verified successfully.

CODING OF SEQUENTIAL CIRCUITS USING HDL

EXP NO. : 14

DATE :

AIM:

To write Verilog programs for sequential circuits.

SOFTWARE REQUIRED:

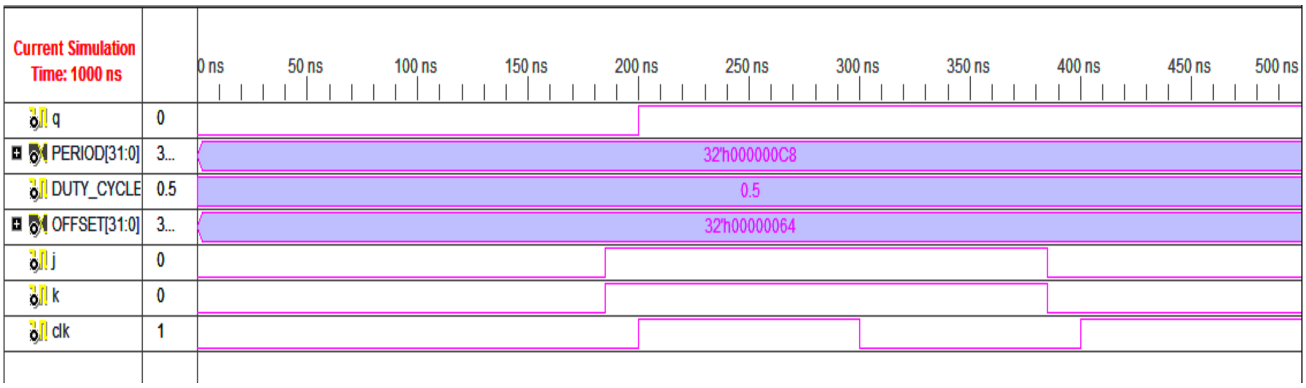
1. Xilinx ISE 10.1
2. Simulation Tools
   a. Verilog-Xilinx Model Sim

PROGRAM:

JK FLIP FLOP

```
module jk( j,k,clk,q);
input j,k,clk;
output reg q;
initial q=1'b0;
always@ (posedge clk)
begin
case({j,k})
2'b00:q=q;
2'b01:q=0;
2'b10:q=1;
2'b11:q= q;
end case
end
endmodule
```

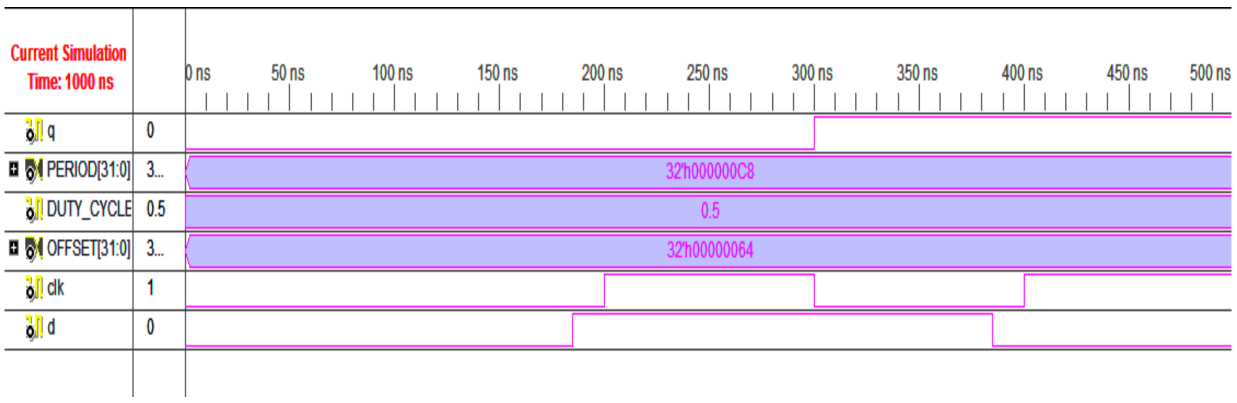
OUTPUT:



D FLIP FLOP

```
module dff( clk,d,q);
input clk,d;
output reg q;
always@ (negedgeclk)
q=d;
endmodule
```

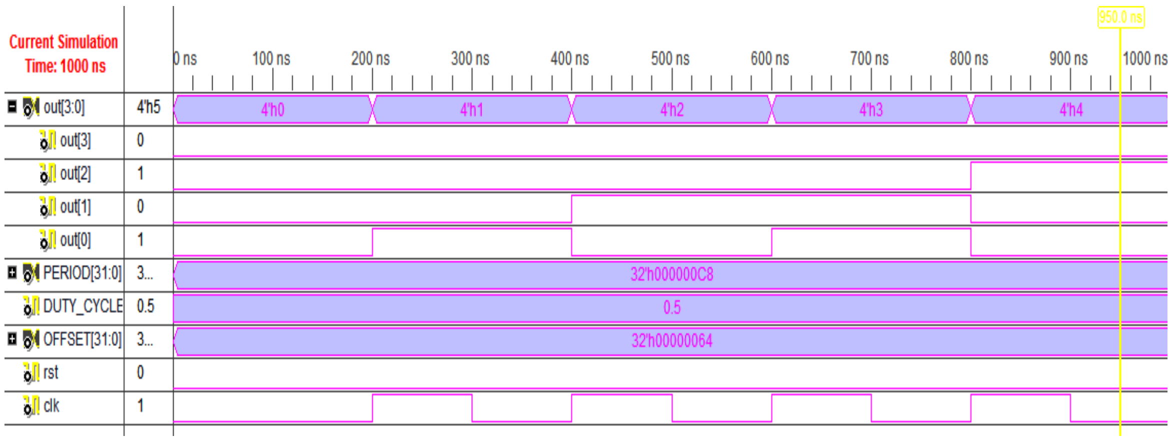
OUTPUT:



MOD-10 COUNTER

```
module mod12(rst,clk,out);
input rst,clk;
output reg[3:0]out=4'b0000;
always@ (posedge clk)
begin
if(rst==1/out==4'b1011)
out=4'b0000;
else
out=out+4'b0001;
end
endmodule
```

OUTPUT:



RESULT:

Thus, the simulation of Sequential circuits was verified successfully.