E.G.S.PILLAY ENGINEERING COLLEGE, NAGAPATTINAM

ELECTRONICS AND COMMUNICATION ENGINEERING

LAB MANUAL

2301GEX05

APPLIED DIGITAL LOGIC AND DESIGN LAB

PREPARED BY
Dr.S.SENTHILKUMAR
ASST.PROFESSOR
ECE DEPT

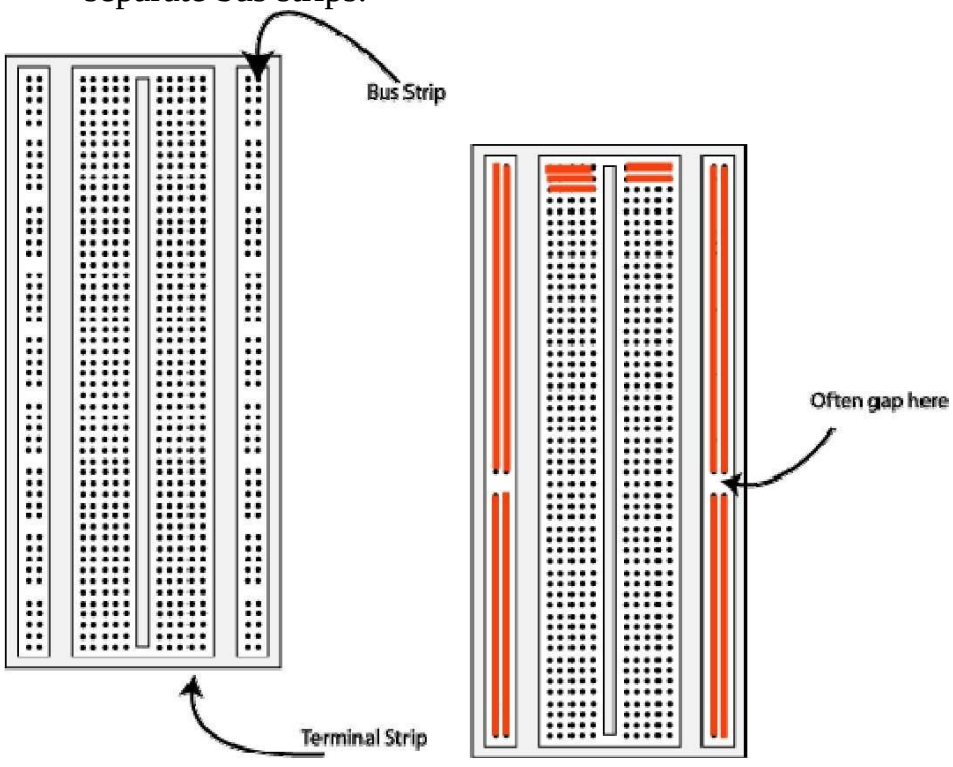
APPLIED DIGITAL LOGIC AND DESIGN

INTRODUCTION

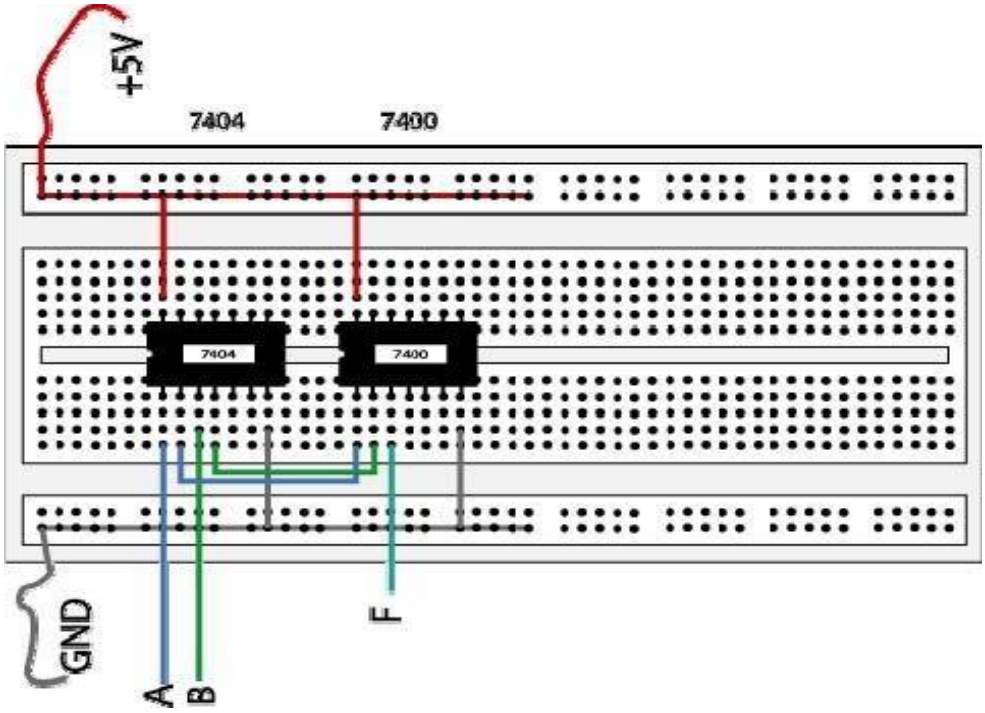
The Breadboard

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

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



The 5V supply **MUST NOT BE EXCEEDED** since this will damage the ICs (Integrated circuits) used during the experiments. Incorrect connection of power to theICscouldresultinthemexplodingorbecomingveryhot-withthe**possible**



DO'S

1. Be regular to the lab, Follow Proper Dress Code &Maintain Silence.
2. Identity the Different Leads or Pins of the IC before making connection.
3. Know the Biasing Voltage required for different families of IC's and connect the Power Supply Voltage and Ground terminals to the Respective pins of the IC's.
4. Know the current and Voltage Rating of the IC's before using them in the experiment.
5. Avoid unnecessary talking while doing the experiment.
6. Handle the IC Trainer Kit Properly.
7. Keep the Table Clean Arrange the chairs/stools and equipment Properly before leaving the lab
8. After the completion of the experiments Switch off the Power supply and return the apparatus

DON'T'S

1. Do not exceed the voltage Rating.
2. Do not inter change the IC's while doing the experiment.
3. Avoid loose connections and short circuits.
4. Do not throw the connecting wires to floor.
5. Do not come late to the lab.
6. Do not operate IC trainer Kits unnecessarily.

2301GEX05	APPLIEDDIGITALLOGICANDDESIGN Common to B.E-CSE, B.Tech-IT ,CSBS and AIDS	L	T	P	C
		3	0	2	4

PREREQUISITE: Basic mathematic skills

COURSEOBJECTIVES:

1. To present the fundamentals of digital circuits and simplification methods.
2. Topracticethedesignofvariouscombinationalandsequentialdigitalcircuitsusinglogicgates.
3. To introduces mi conductor memories and programmable logic devices.
4. To practice the HDL programming for combinational and sequential circuits.

COURSEOUTCOMES:

On the successful completion of the course, students will be able to

CO1: Use Boolean algebra ,K-map and tabulation method to simplify Boolean functions.
CO2: Construct different combinational circuits using logic gates.
CO3: Develop different sequential circuits using logic gates and flip flops.
CO4: Compare different Semi conductor memory devices.
CO5: Build programmable devices using logic gates.
CO6: Develop Verilog program for combinational and sequential circuits.

Cos Vs Pos MAPPING:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	2	1						2	1		1
CO2	3	2	1						2	1		1
CO3	3	2	1						2	1		1
CO4	3	2	1						2	1		1
CO5	3	2	1						2	1		1
CO6	3	2	1		3				2	1		1

Cos Vs PSOs MAPPING

COs	PSO1	PSO2
CO1	3	
CO2	3	
CO3	3	
CO4	3	
CO5	3	
CO6	3	

COURSECONTENTS:

MODULEI	BOOLEAN ALGEBRA AND LOGIC GATES	12Hours

Review of Number system–Boolean expression and minimization–Logic Gates and its implementation-Simplification of Boolean Functions using Boolean algebra,Karnaugh Map and Tabulation Method.

UNITII	COMBINATIONAL LOGIC	15Hours

Combinational Circuits–Analysis and Design Procedures–Circuits for Arithmetic Operations, Code Conversion–Decoders/Encoders–Multiplexers/De multiplexers-Parity generators/checkers-Magnitude Comparator.

UNIT III	SEQUENTIAL CIRCUITS	12Hours

Sequential logic-Basic latch-Flip-flops(SR,D,JK,TandMaster-Slave)-Counters-Ripplecounters-BCDand Binary-Synchronouscounters,Registers-Shiftregisters-Registers,Hazards

UNIT IV	MEMORY AND PROGRAMMABLE LOGIC	9Hours

Classification of memories (RAM, ROM ,PROM,EPROM, EEPROM)-Programmable Logic Devices(PLA,PAL,FPGA)-Implementation of circuits using ROM,PLA, PAL.

UNIT V	Verilog HDL modeling	12Hours

3types of Verilog modeling(gate-level,dataflow, and behavioral)-Verilog programming for combinational And sequential circuits.

List of Lab experiments

1. Verification of Boolean Theorems using basic gates
2. Design and implementationofhalfadder,halfsubtractor,fulladderandfullsubtractor
3. Design and implementation of code converters
4. Design and implementation of multiplexer and de-multiplexer
5. Design and implementation parity generator/checker
6. Design and implementation counters
7. Design and implementation shift register
8. Develop and simulation of Verilog program for combinational circuits
9. Develop and simulation of Verilog program for sequential circuits

Hardware/software requirement

1. Digital trainer kit 10Nos
2. Adequate numbers of IC‟s
3. Xilinx ISE(or)Altera QuartusII software

TOTAL:60 HOURS

REFERENCES:

1.MorrisManoandMichaelD.Ciletti,"DigitalDesign",5thedition,PrenticeHallofIndia,2012

2.SamirPalnitkar,"VerilogHDL",2ndEdition,PearsonEducation,2003

3.https://archive.nptel.ac.in/courses/108/105/108105132/(LinkforNPTEL/SWAYAM/MOOCCourses)

4.https://www.vlab.co.in/broad-area-electronics-and-communications(Linkformoderntoolusage)

STUDY OF LOGIC GATES

Ex. No:
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.	CONNECTING WIRES	As per Required	

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:

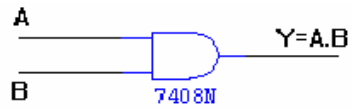
The NOR gate is a contraction of OR-NOT. The output is high when both inputs are low. The output is low when one or both inputs are high.

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.

AND GATE:

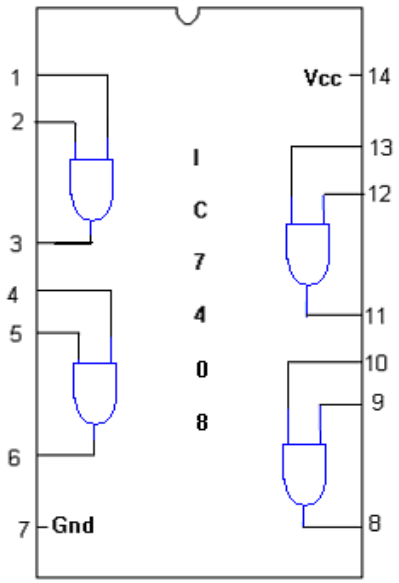
SYMBOL:



TRUTH TABLE

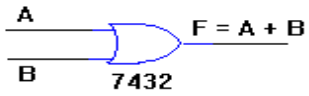
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

PIN DIAGRAM:



OR GATE:

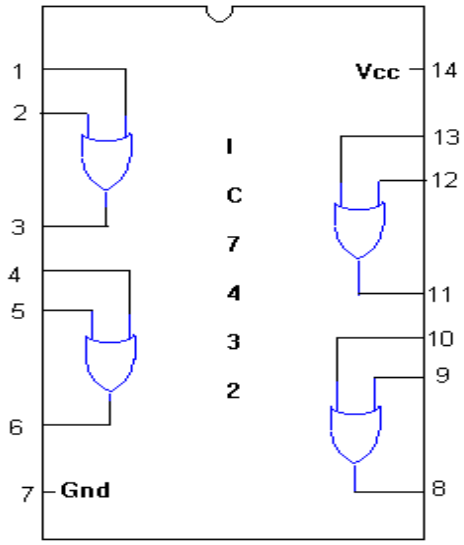
SYMBOL :



TRUTH TABLE

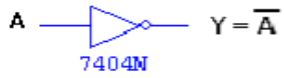
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

PIN DIAGRAM :

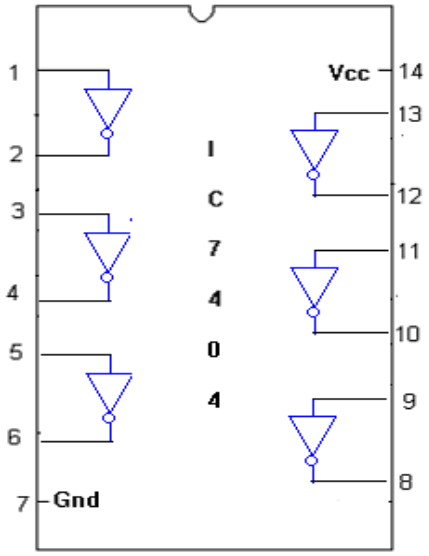


NOT GATE:

SYMBOL:



PIN DIAGRAM:

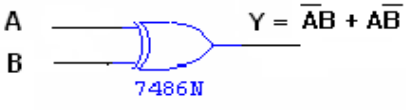


TRUTH TABLE :

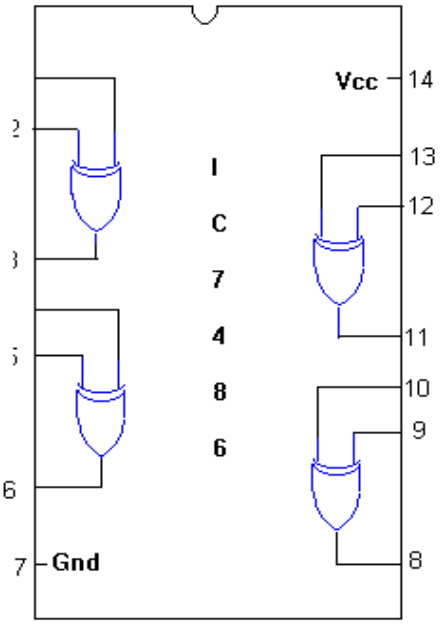
A	$\overline{A}$
0	1
1	0

X-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:

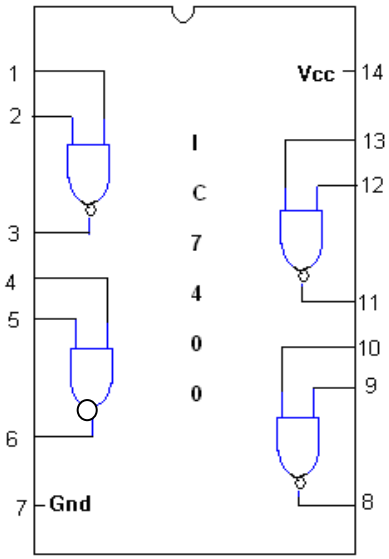
SYMBOL:

7400

TRUTH TABLE

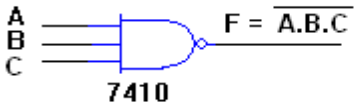
A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

PIN DIAGRAM:

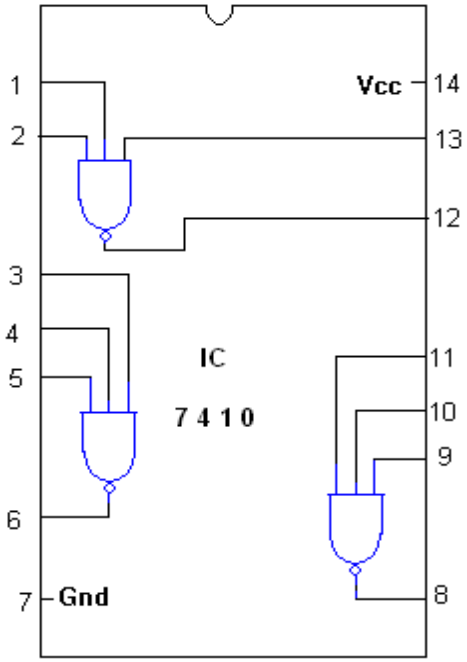


3-INPUT NAND GATE :

SYMBOL :



PIN DIAGRAM :

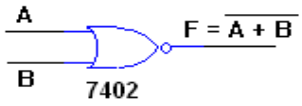


TRUTH TABLE

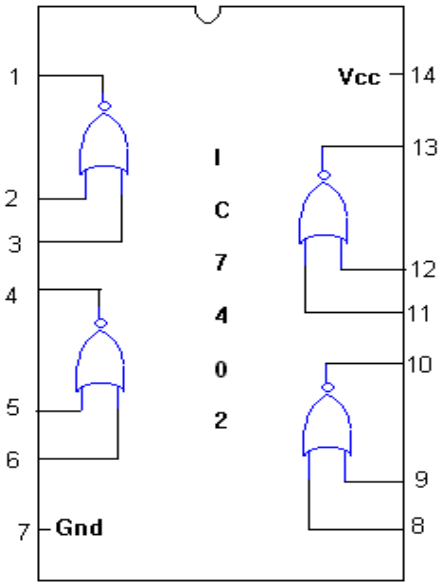
A	B	C	$\overline{A.B.C}$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

NOR GATE:

SYMBOL :



PIN DIAGRAM :



TRUTH TABLE

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

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 AND, OR, NOT, XOR, NOR and NAND gates were studied and their truth table were verified successfully.

VERIFICATION OF BOOLEAN THEOREMS USING DIGITAL LOGIC GATES

Ex. No. :

Date :

AIM:

To verify the Boolean Theorems using logic gates.

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.	CONNECTING WIRES	As per Required	

BOOLEAN THEOREM:

Theorem : 1. x + x = x &

x . x = x

2. x + 1 = 1 ; x . 0 = 0
3. (x')' = x (Involution)
4. Associative

x + (y + z) = (x + y) + z &

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

5. De Morgan's (x + y)' = x' . y'

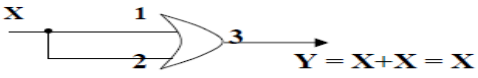
(x . y)' = x' + y'

6. Absorption x + x.y = x &

x . (x + y) = x

CIRCUIT DIAGRAM:

THEOREM : 1



IC 7432

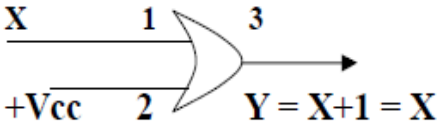
IC 7408

Truth Table:

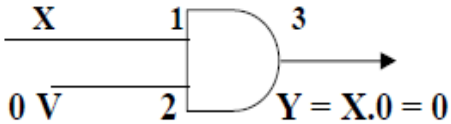
X	X	Y = X+X
0	0	0
1	1	1

X	X	Y = X.X
0	0	0
1	1	1

THEOREM: 2



IC 7432



IC 7408

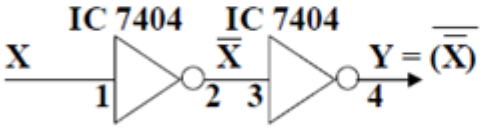
Truth Table

X	1	Y=X+1=1
0	1	1
1	1	1

X	0	Y=X.0=0
0	0	0
1	0	0

THEOREM: 3 (INVOLUTION THEOREM)

Truth Table:



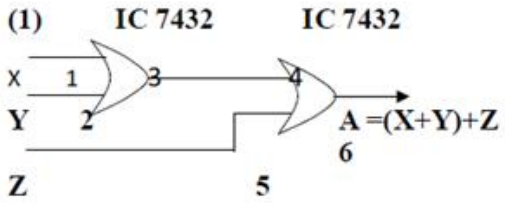
X	$\overline{X}$	$Y = \overline{(\overline{X})}$
0	1	0
1	0	1

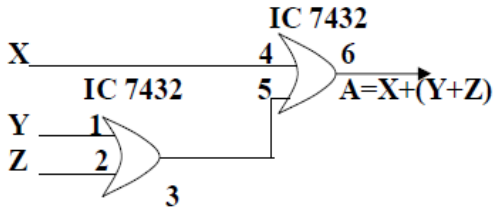
THEOREM:4 (ASSOCIATIVE)

Truth Table:

X	Y	Z	X+Y	A
0	0	0	0	0
0	1	0	1	1
1	0	1	1	1
1	1	1	1	1

(1)

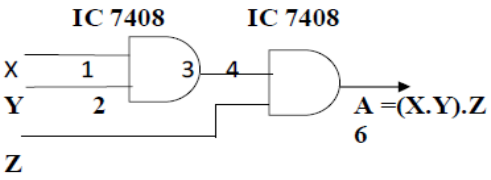




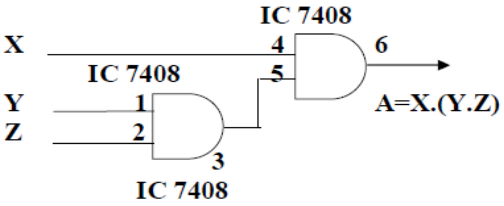
X	Y	Z	Y+Z	A
0	0	0	0	0
0	1	0	1	1
1	0	1	1	1
1	1	1	1	1

(2) **(X.Y).Z =X.(Y.Z)**

Truth Table:



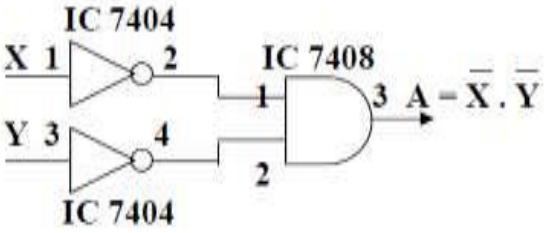
X	Y	Z	X.Y	A
0	0	0	0	0
0	1	0	0	0
1	0	1	0	0
1	1	1	1	1



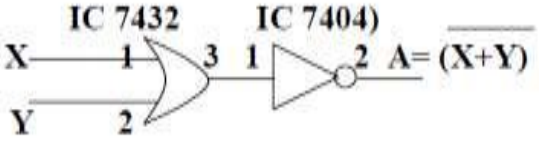
X	Y	Z	Y.Z	A
0	0	0	0	0
0	1	0	0	0
1	0	1	0	0
1	1	1	1	1

THEOREM: 5 DE-MORGAN'S LAW:

(i) X . Y = (X+Y))



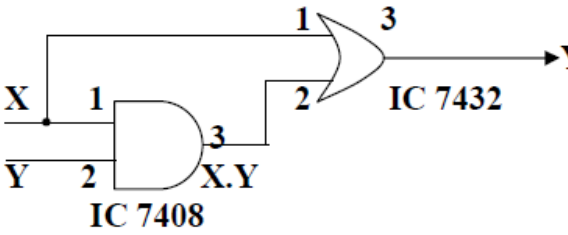
X	Y	$\overline{X}$	$\overline{Y}$	A=$\overline{X}$. $\overline{Y}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0



Truth Table:

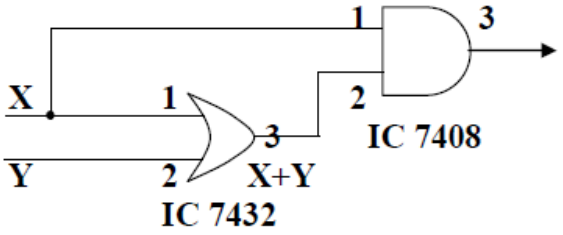
X	Y	X+Y	A=$\overline{(X+Y)}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

THEOREM: 6 (ABSORPTION)



Truth Table:

X	Y	X+Y	Y = X + X.Y	X
0	0	0	0	0
0	1	0	0	0
1	0	0	1	1
1	1	1	1	1



Truth Table:

X	Y	X+Y	Y = X.(X+Y)	X
0	0	0	0	0
0	1	0	0	0
1	0	0	1	1
1	1	1	1	1

PROCEDURE:

1. The connections are made as per the circuit diagram.
2. Give the logical inputs as per the truth table.
3. The corresponding output is verified with their truth table.

RESULT:

Thus the basic Boolean algebra and Demorgon's theorem were verified using basic logic gates its corresponding truth tables successfully.

DESIGN OF ADDER AND SUBTRACTOR

Ex. No. :

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.	CONNECTING WIRES	As per Required	

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:

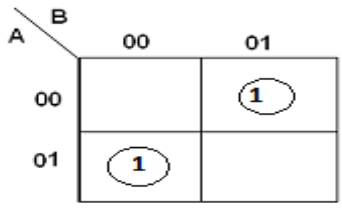
The full subtractor is a combination of X-OR, AND, OR, NOT Gates. In a full subtractor the logic circuit should have three inputs and two outputs. The two half subtractor put together gives a full subtractor .The first half subtractor will be C and A B. The output will be difference output of full subtractor.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

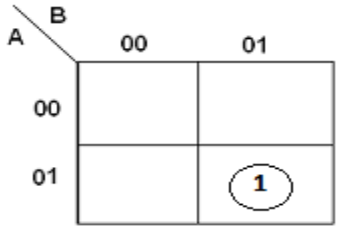
A	B	CARRY	SUM
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

K-Map for SUM:



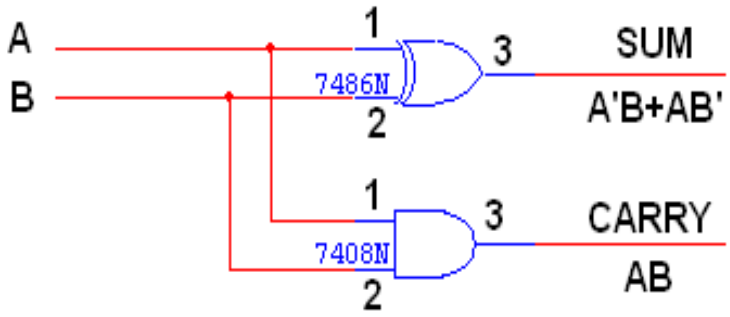
SUM = A'B + AB'

K-Map for CARRY:



CARRY = AB

LOGIC DIAGRAM:

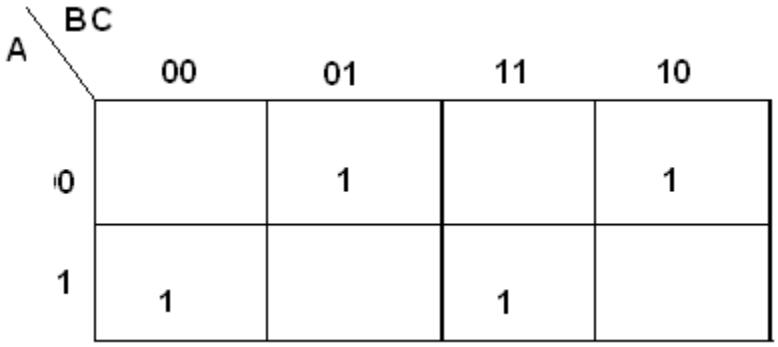


FULL ADDER

TRUTH TABLE:

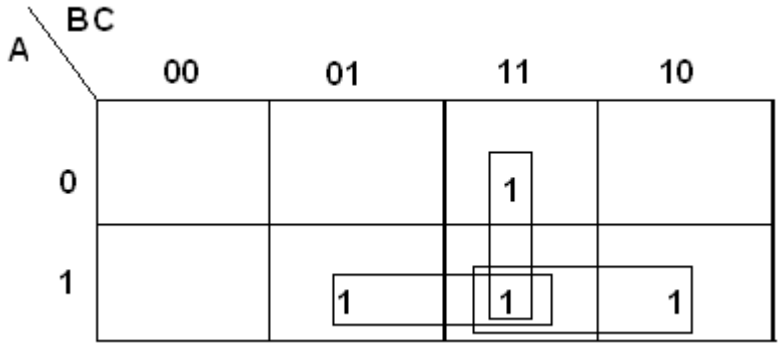
A	B	C	CARRY	SUM
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

K-Map for SUM:



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

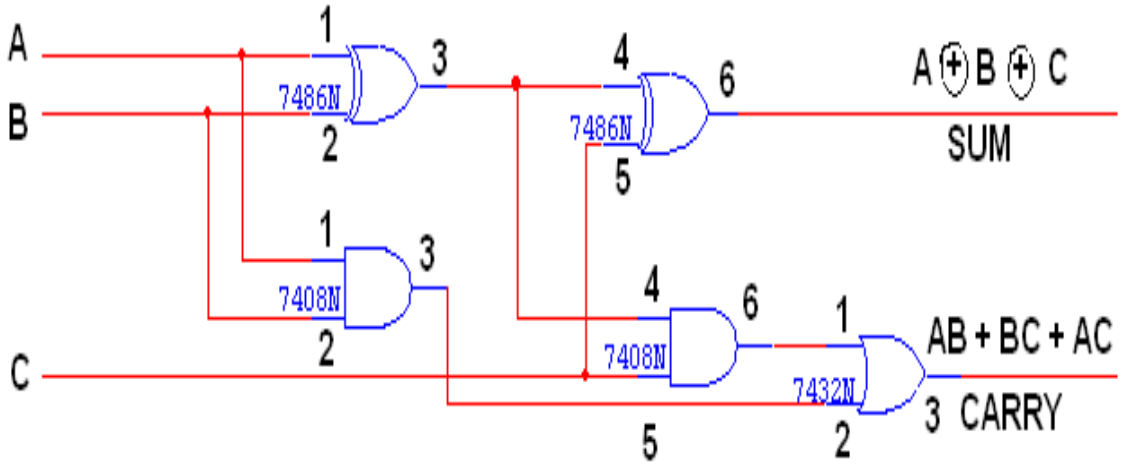
K-Map for CARRY:



CARRY = AB + BC + AC

LOGIC DIAGRAM:

FULL ADDER USING TWO HALF ADDER

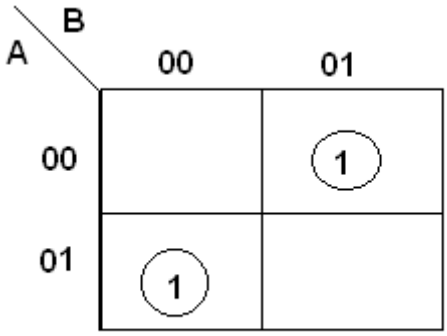


HALF SUBTRACTOR

TRUTH TABLE:

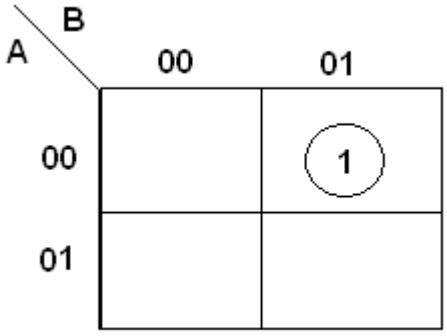
A	B	BORROW	DIFFERENCE
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

K-Map for DIFFERENCE:



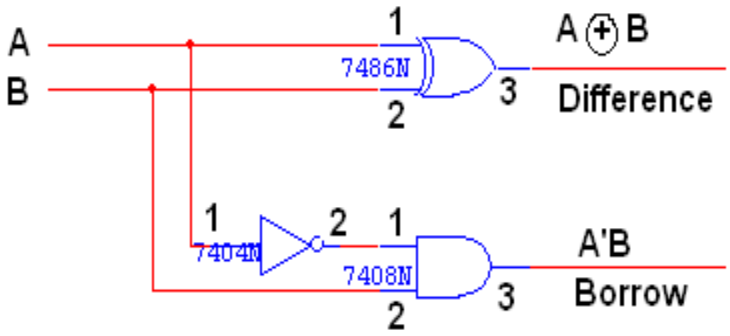
DIFFERENCE = A'B + AB'

K-Map for BORROW:



BORROW = A'B

LOGIC DIAGRAM:

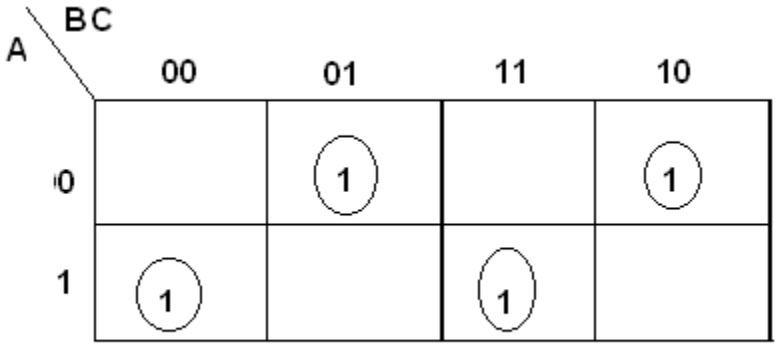


FULL SUBTRACTOR

TRUTH TABLE:

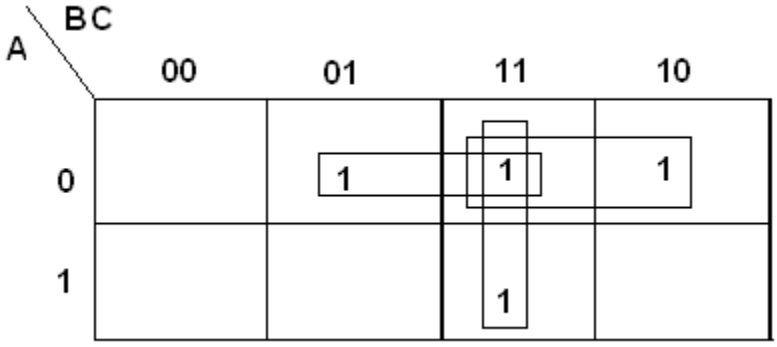
A	B	C	BORROW	DIFFERENCE
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

K-Map for Difference:



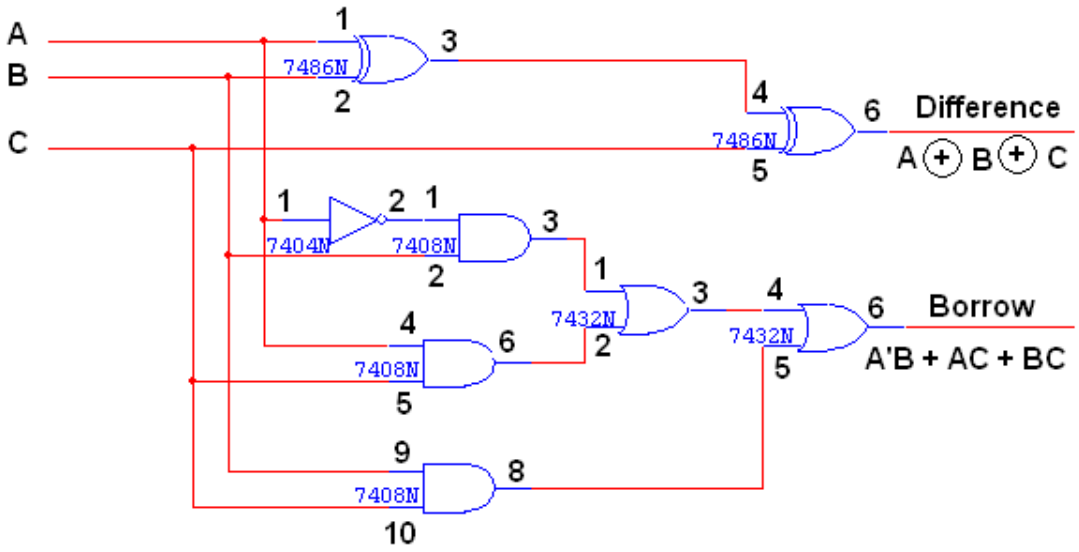
Difference = A'B'C + A'BC' + AB'C' + ABC

K-Map for Borrow:

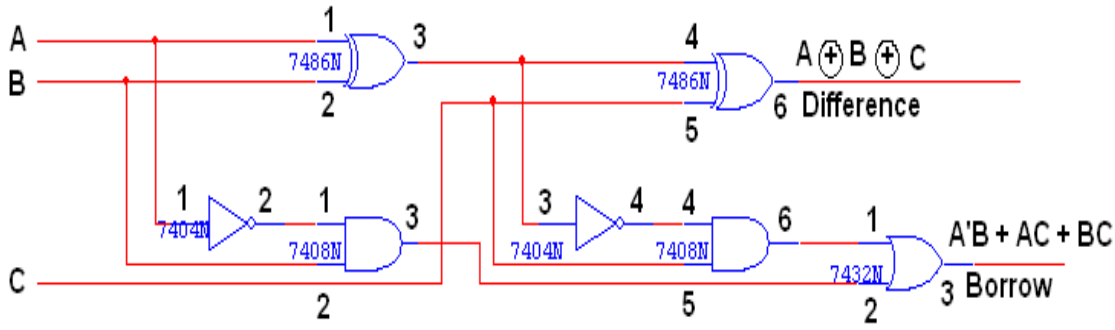


Borrow = A'B + BC + A'C

LOGIC DIAGRAM:



FULL SUBTRACTOR USING TWO HALF SUBTRACTOR:



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:

Thus Half adder, Full adder, half subtractor and Full subtractor were designed and their truth tables were verified successfully.

DESIGN AND IMPLEMENTATION OF CODE CONVERTORS

Ex. No :

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 Code Converters

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.	CONNECTING WIRES	As per Required	

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.

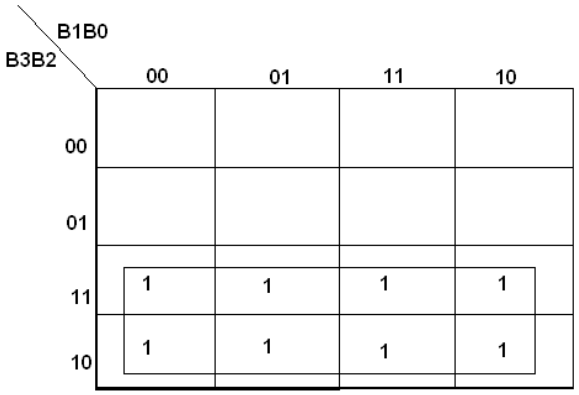
A two-level logic diagram may be obtained directly from the Boolean expressions derived by the maps. These are various other possibilities for a logic diagram that implements this circuit. Now the OR gate whose output is C+D has been used to implement partially each of three outputs.

BINARY TO GRAY CODE CONVERTOR

TRUTH TABLE:

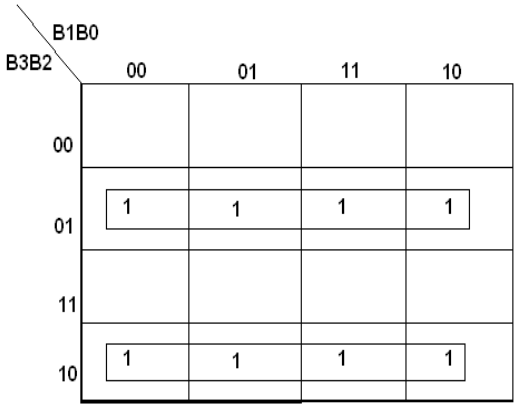
Binary input				**Gray code output**			
B3	**B2**	**B1**	**B0**	**G3**	**G2**	**G1**	**G0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**1**	**0**	**0**	**0**	**1**
0	**0**	**1**	**0**	**0**	**0**	**1**	**1**
0	**0**	**1**	**1**	**0**	**0**	**1**	**0**
0	**1**	**0**	**0**	**0**	**1**	**1**	**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:



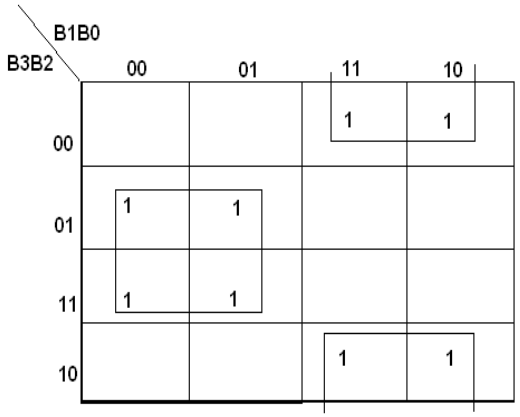
$G_3 = B_3$

K-Map for G_2:



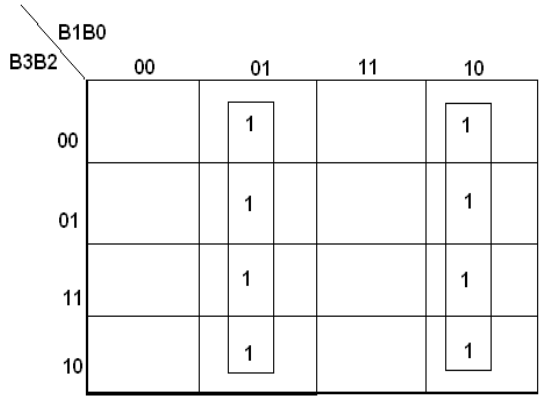
G2 = B3⊕B2

K-Map for G_1:



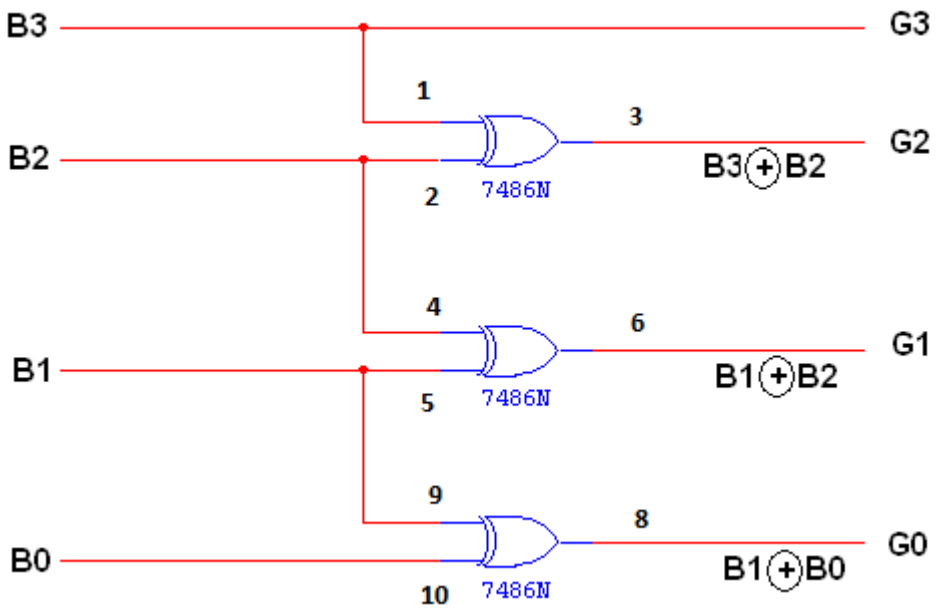
G1 = B1⊕B2

K-Map for G_0:



$G0 = B1 \oplus B0$

LOGIC DIAGRAM:

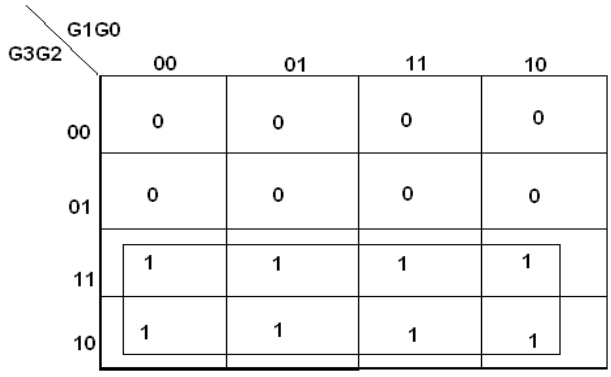


GRAY CODE TO BINARY CONVERTOR

TRUTH TABLE:

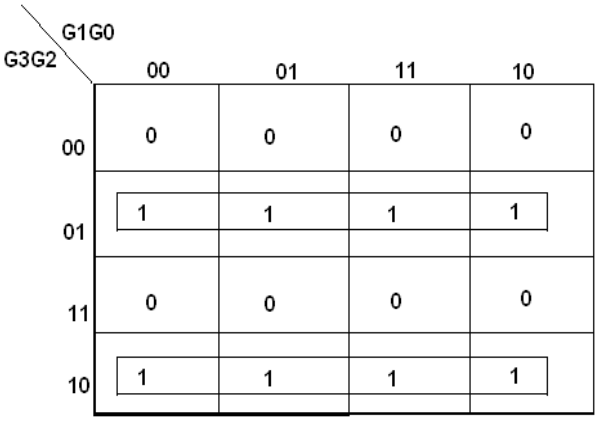
Gray Code				Binary Code			
G3	**G2**	**G1**	**G0**	**B3**	**B2**	**B1**	**B0**
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

K-Map for B_3:



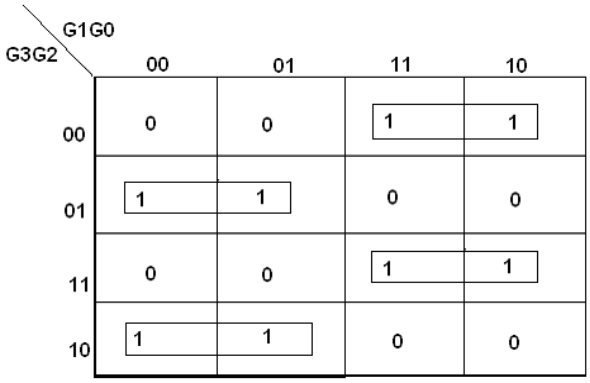
B3 = G3

K-Map for B_2:



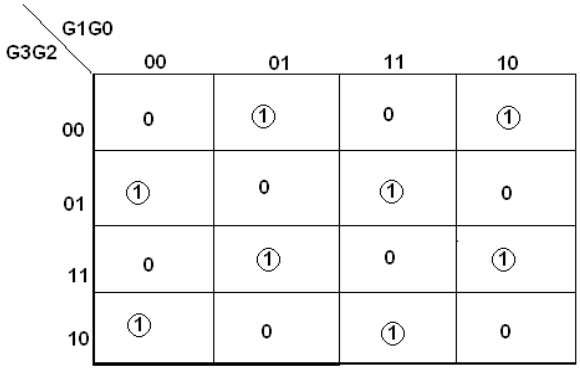
$B2 = G3 \oplus G2$

K-Map for B_1:



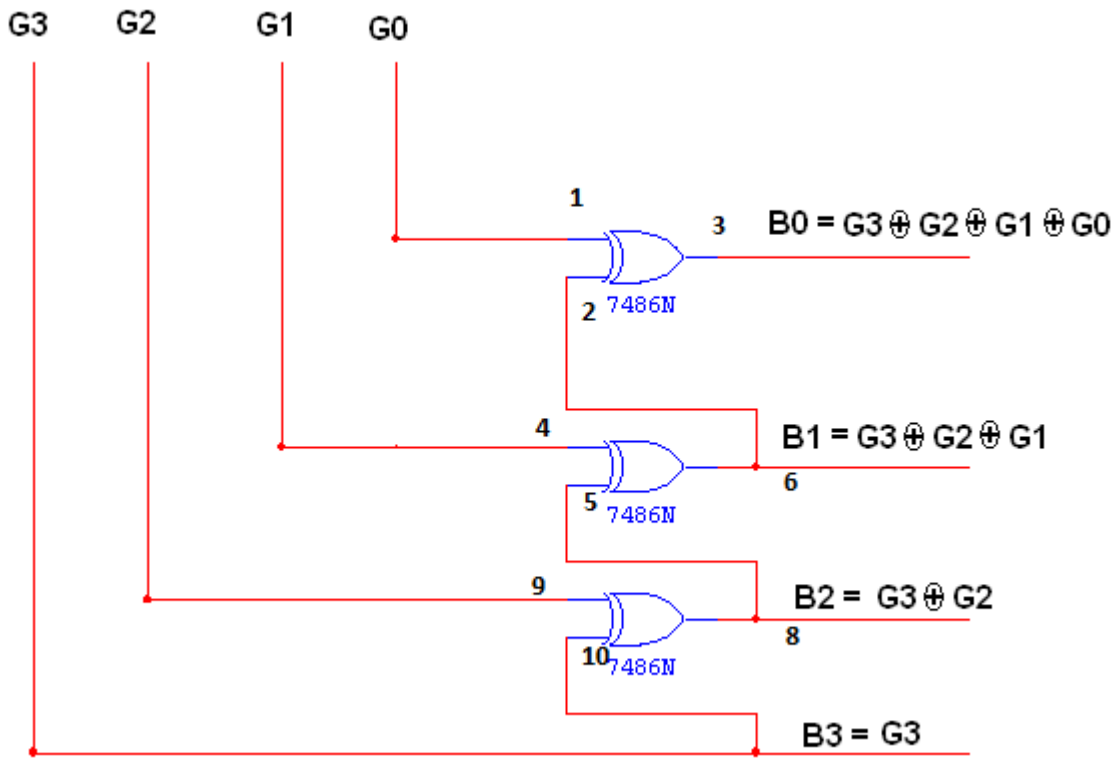
$B1 = G3 \oplus G2 \oplus G1$

K-Map for B_0:



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

LOGIC DIAGRAM:

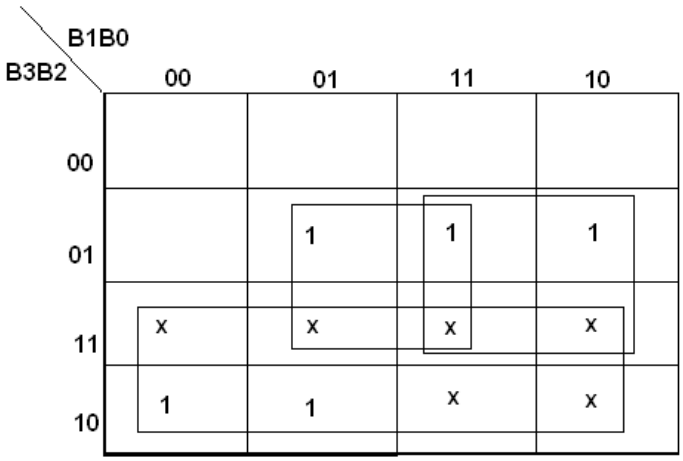


BCD to Excess - 3

Truth Table:

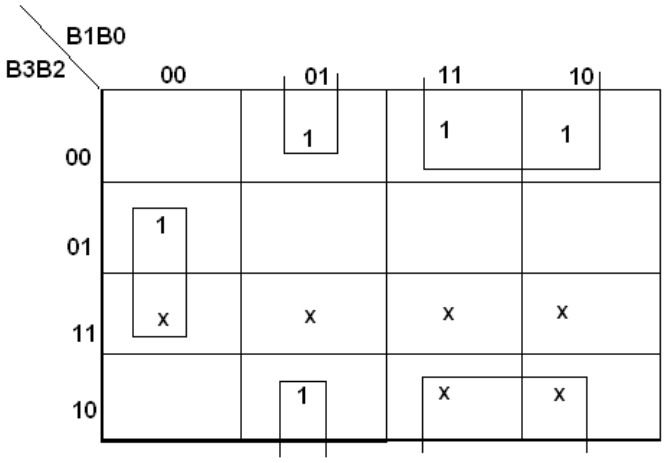
INPUT				OUTPUT			
B3	**B2**	**B1**	**B0**	**E3**	**E2**	**E1**	**E0**
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

K Map for E_3:



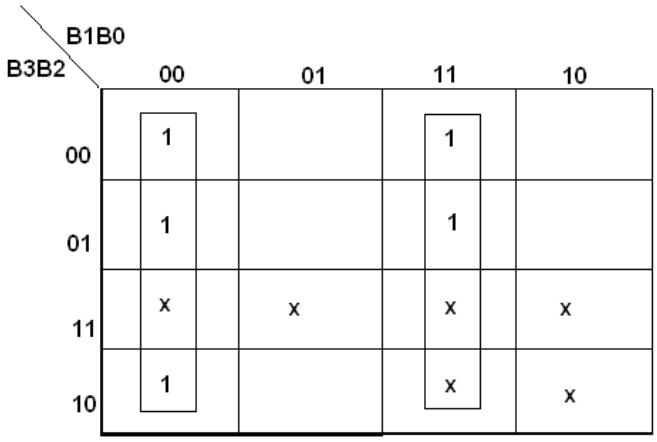
$$E_3 = B_3 + B_2 (B_1 + B_0)$$

K Map for E_2:



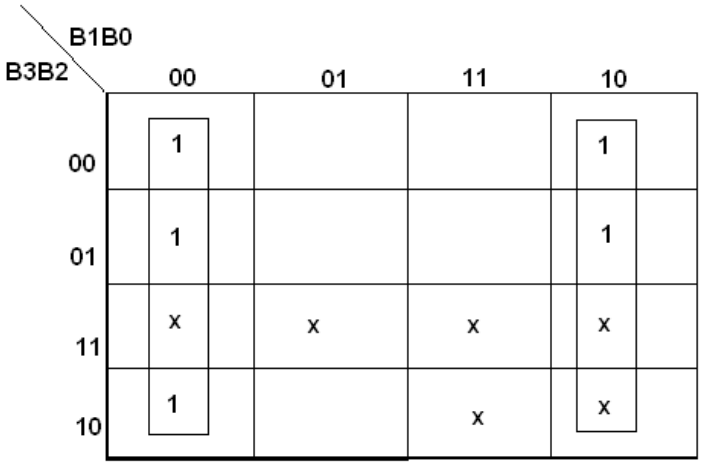
$$E2 = B2 \oplus (B1 + B0)$$

K Map for E_1:



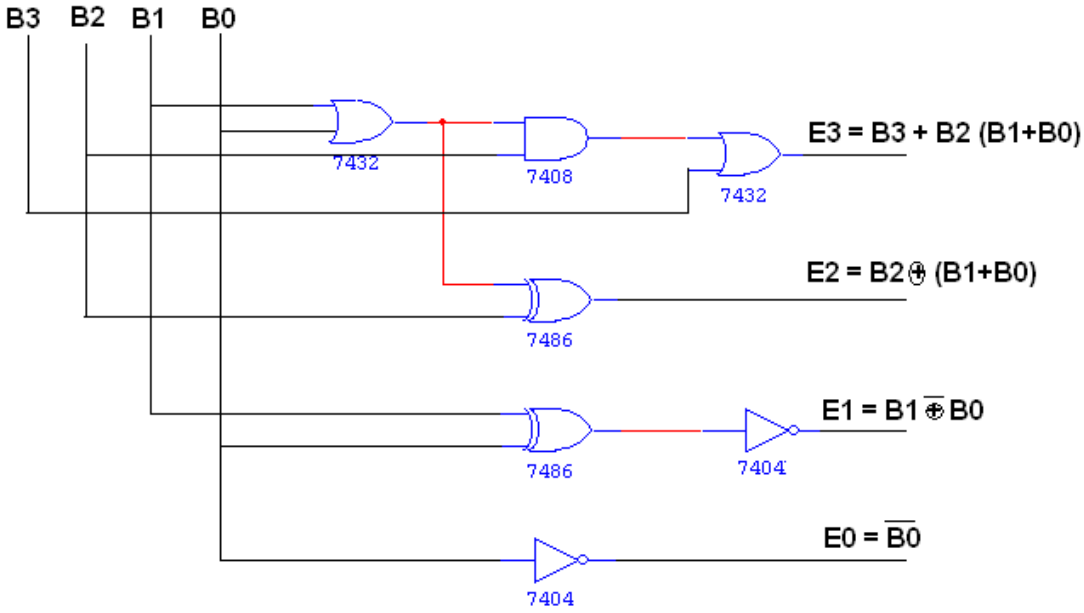
$$E1 = B1 \bar{\oplus} B0$$

K Map for E_0:



$E0 = \overline{B0}$

Logic Diagram



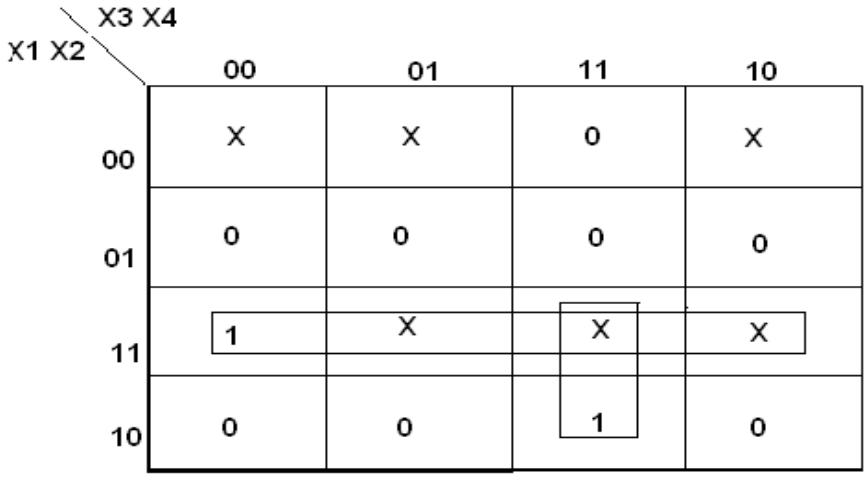
Excess – 3 to BCD
Truth Table:

INPUT				OUTPUT			
E3	**E2**	**E1**	**E0**	**B3**	**B2**	**B1**	**B0**
0	0	0	0	X	X	X	X
0	0	0	1	X	X	X	X
0	0	1	0	X	X	X	X
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
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:

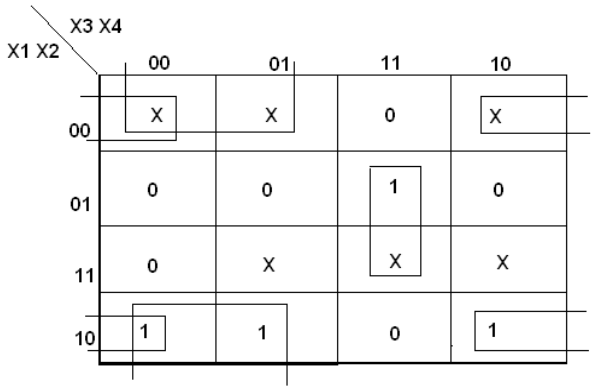
EXCESS-3 TO BCD CONVERTOR

K-Map for A:



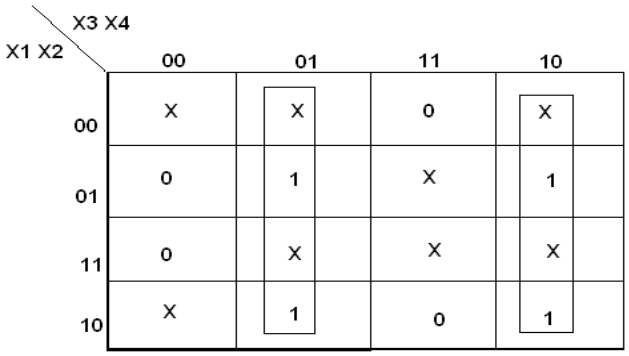
A = X1 X2 + X3 X4 X1

K-Map for B:



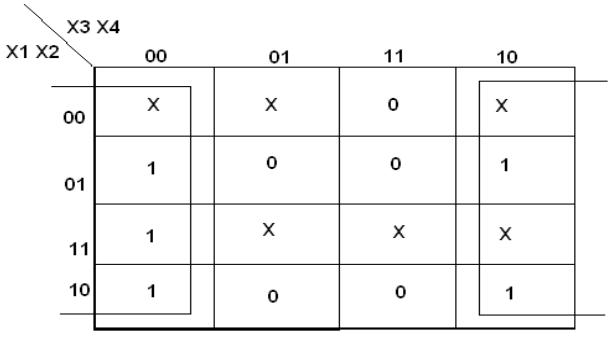
$B = X2 \oplus (\overline{X3} + \overline{X4})$

K-Map for C:

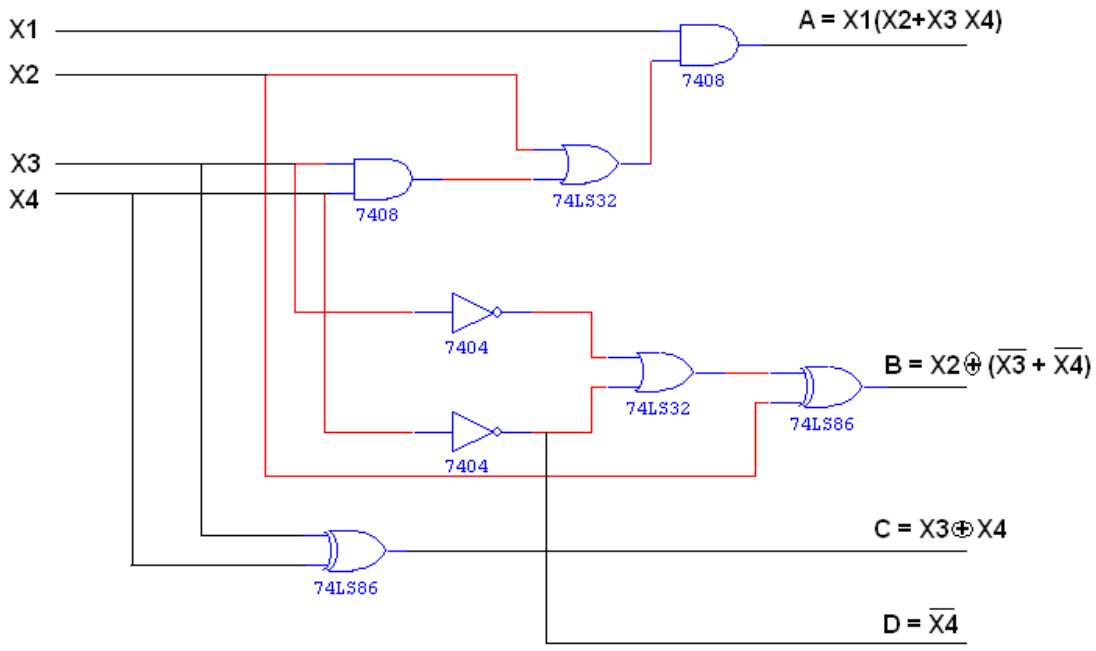


$C = X3 \oplus X4$

K-Map for D:



$D = \overline{X4}$



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 Gray to binary, binary to gray, Excess – 3 to BCD and BCD to Excess – 3 code converters were designed and their truth tables were verified successfully.

DESIGN AND IMPLEMENTATION OF MULTIPLEXER AND DEMULTIPLEXER

Ex. No. :

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
4.	IC TRAINER KIT	-	1
5.	CONNECTING WIRES	As per Required	

THEORY:

MULTIPLEXER:

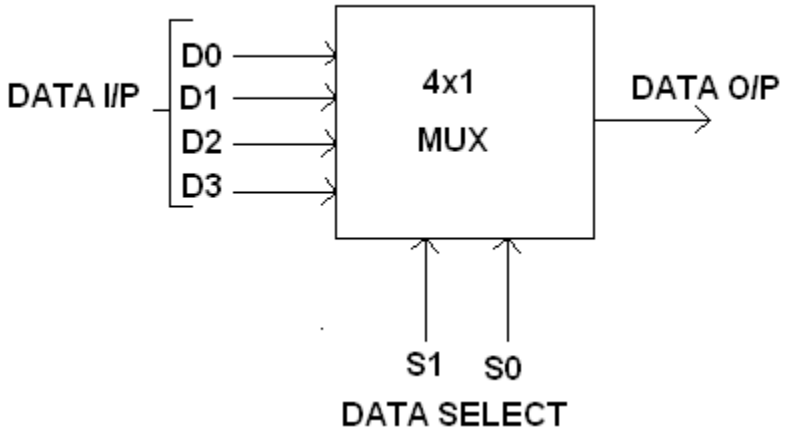
Multiplexer means transmitting a large number of information units over a smaller number of channels or lines. A digital multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line. The selection of a particular input line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

DE-MULTIPLEXER:

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:

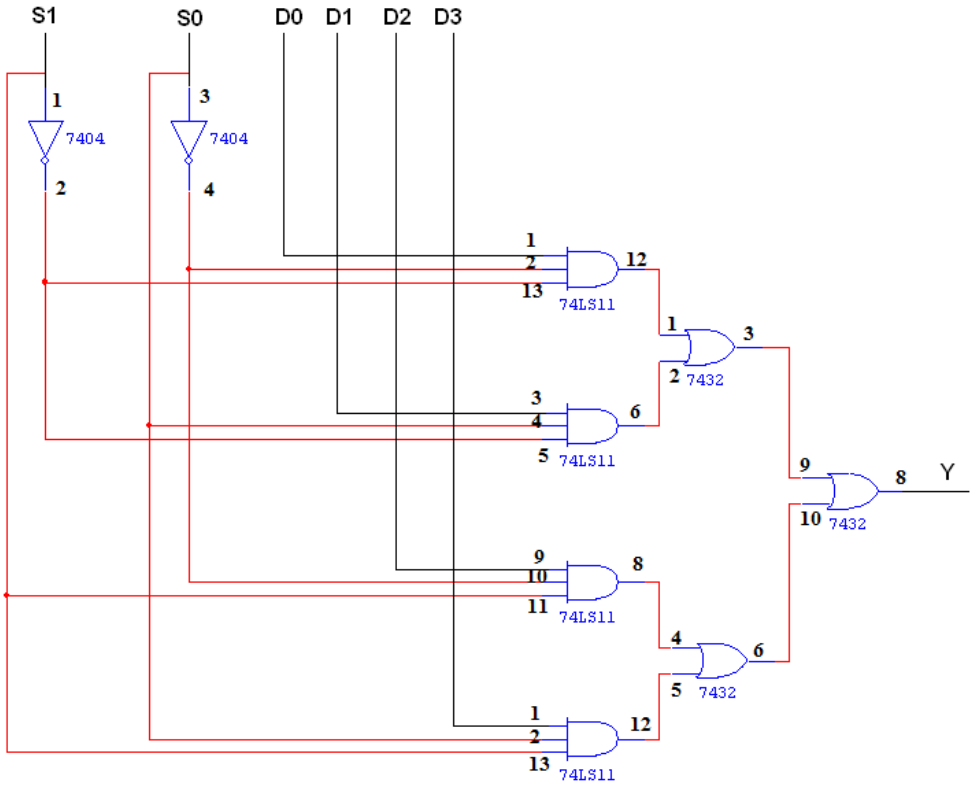


FUNCTION TABLE:

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 S0

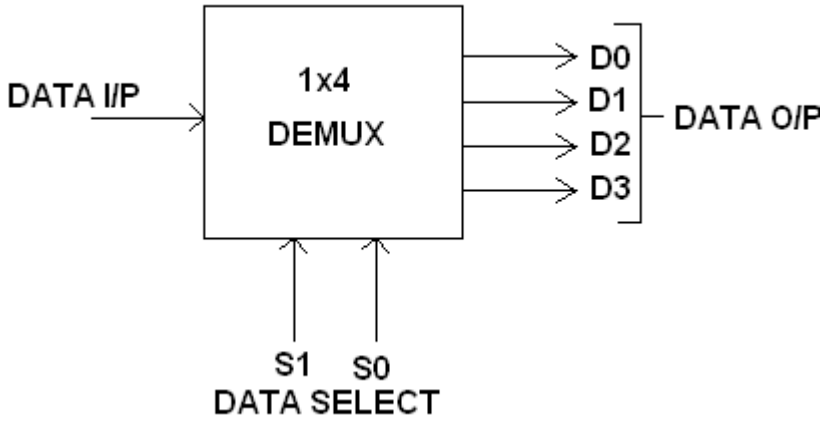
CIRCUIT DIAGRAM FOR MULTIPLEXER:



TRUTH TABLE:

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

BLOCK DIAGRAM FOR 1:4 DEMULTIPLEXER:

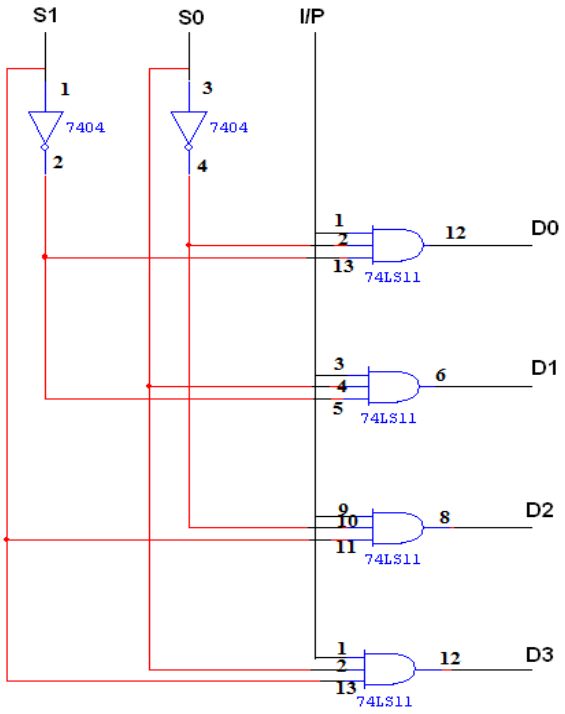


FUNCTION TABLE:

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

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 multiplexer and de-multiplexer circuits were designed and their truth tables were verified successfully.

3 BIT ODD/EVEN PARITY CHECKER /GENERATOR

Ex. No. :

Date :

AIM:

To design and implement 3 - bit odd/even parity checker generator using logic gates.

APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	NOT GATE	IC 7404	1
2.	EX-OR gate	IC 7486	1
3.	IC TRAINER KIT	-	1
4.	CONNECTING WIRES	As per Required	

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.

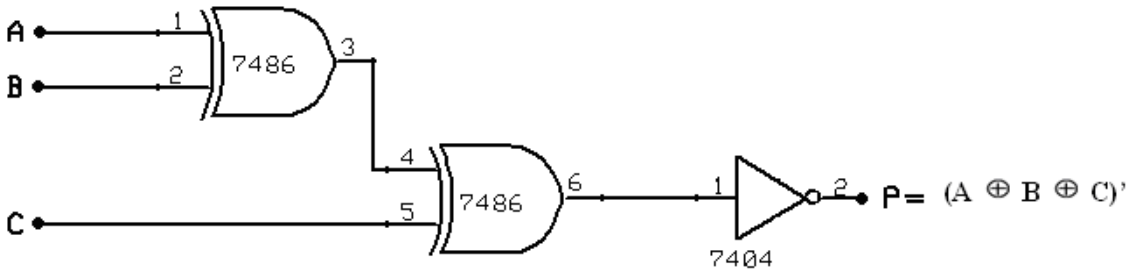
ODD PARITY GENERATOR:

TRUTH TABLE:

S.NO	INPUT (three bit message)			OUTPUT (Odd parity bit)
	A	B	C	P
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0
4	0	1	1	1
5	1	0	0	0
6	1	0	1	1
7	1	1	0	1
8	1	1	1	0

From the truth table the expression for the output parity bit is
P(A,B,C)=Σ(0,3,5,6)
Also written as
P=A'B'C'+A'BC+AB'C+ABC'

Circuit diagram:



ODD PARITY CHECKER:

Truth Table:

S.No	INPUT (four bit message Received)				OUTPUT (Parity error check)
	A	B	C	P	X
1.	0	0	0	0	1
2.	0	0	0	1	0
3.	0	0	1	0	0
4.	0	0	1	1	1
5.	0	1	0	0	0
6.	0	1	0	1	1
7.	0	1	1	0	1
8.	0	1	1	1	0
9.	1	0	0	0	0
10.	1	0	0	1	1
11.	1	0	1	0	1
12.	1	0	1	1	0
13.	1	1	0	0	1
14.	1	1	0	1	0
15.	1	1	1	0	0
16.	1	1	1	1	1

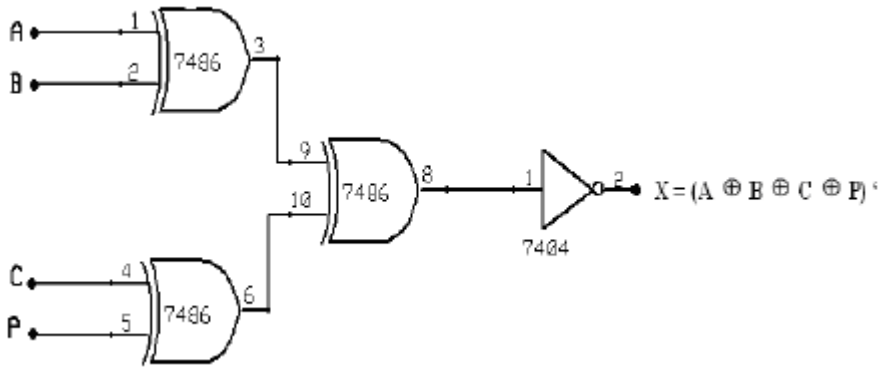
From the truth table the expression for the parity checker bit is

$$X(A,B,C,P)=\Sigma(0,3,5,6,9,10,12,15)$$

The above expression is reduced as

$$X=(A \oplus B \oplus C \oplus P)$$

CIRCUIT 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 the 3 bit ODD / EVEN parity Generator / Checker were designed and their truth tables were verified successfully.

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

Ex.No. :

Date :

AIM:

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

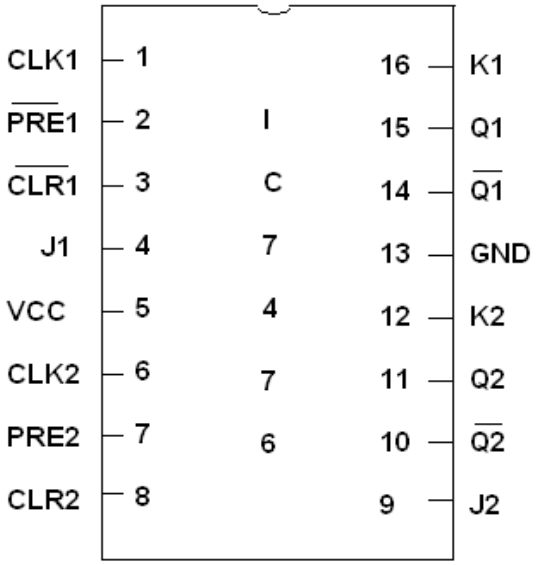
APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	JK FLIP FLOP	IC 7476	2
2.	NAND GATE	IC 7400	1
3.	IC TRAINER KIT	-	1
4.	CONNECTING WIRES	As per Required	

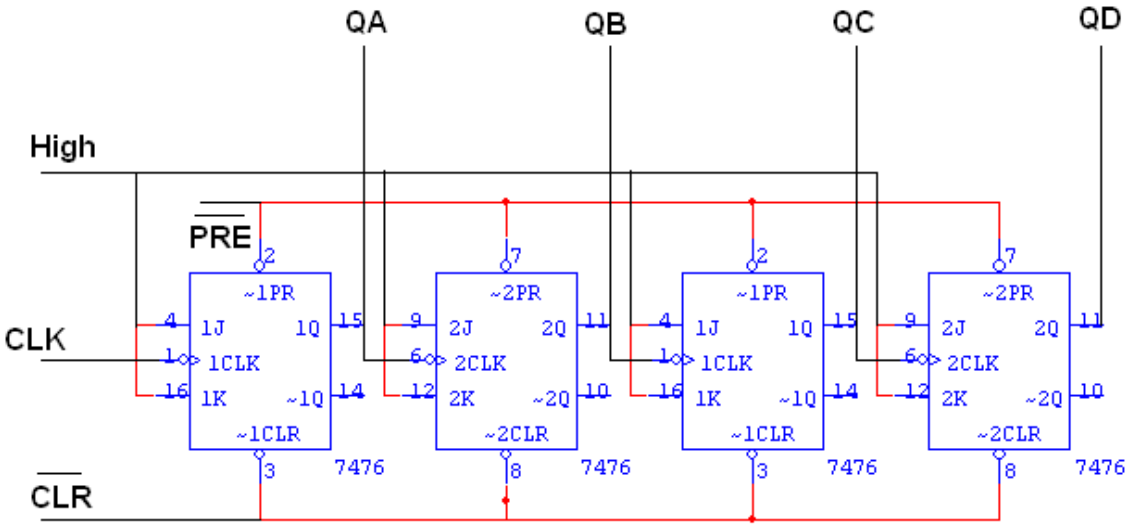
THEORY:

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:



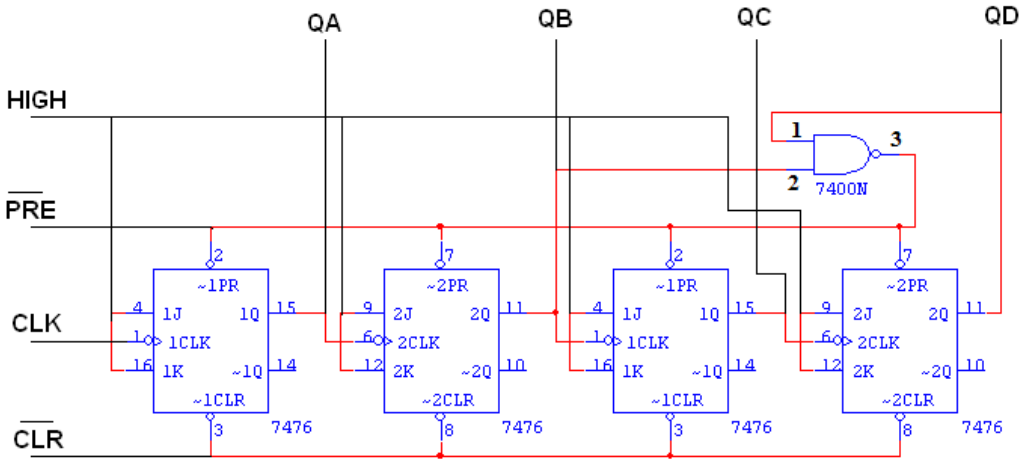
LOGIC DIAGRAM FOR 4 BIT RIPPLE COUNTER:



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

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 4 – bit ripple counter and MOD – 10 counter were designed and their truth tables were verified successfully.

DESIGN AND IMPLEMENTATION OF 3 BIT SYNCHRONOUS UP/DOWN COUNTER

Ex. No. :

Date :

AIM:

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

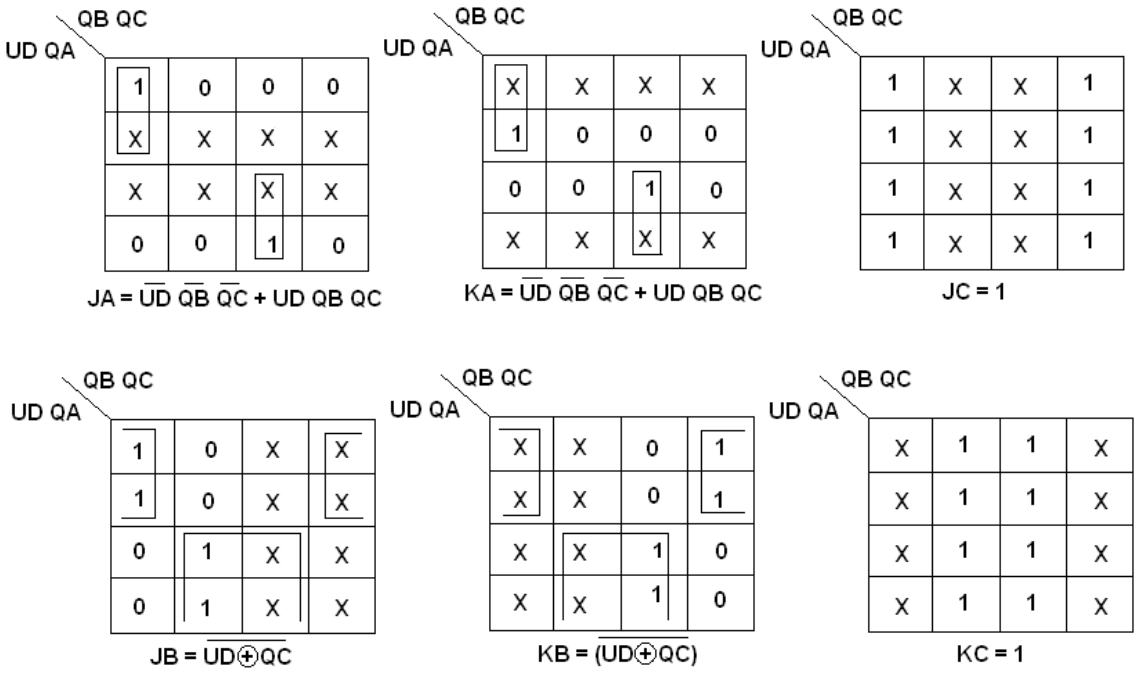
APPARATUS REQUIRED:

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.	CONNECTING WIRES	As per Required	

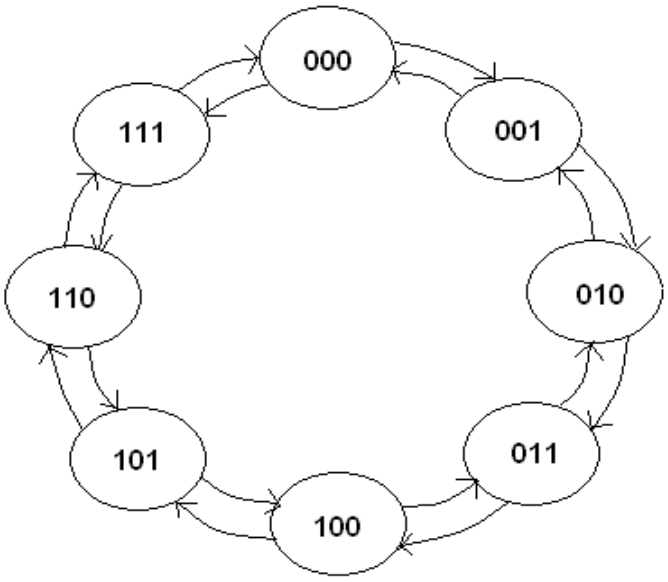
THEORY:

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



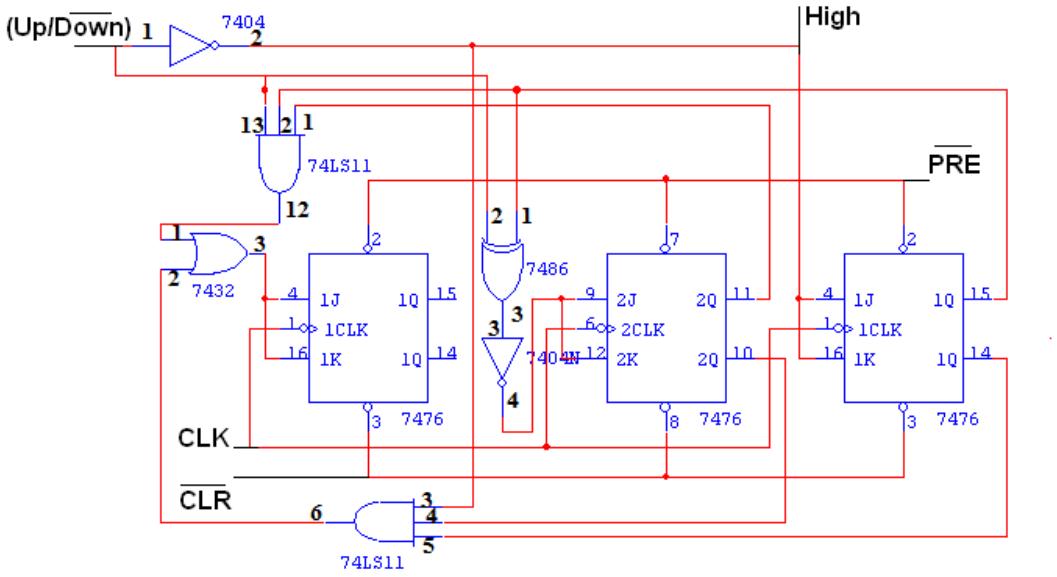
STATE DIAGRAM:



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:

Thus 3 – bit synchronous counter was designed and its truth table was verified successfully.

DESIGN AND IMPLEMENTATION OF SHIFT REGISTERS

Ex. No. :

Date :

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 Shift registers

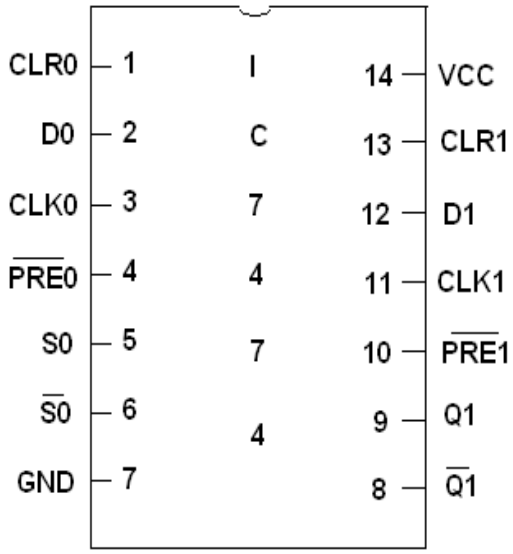
APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	D FLIP FLOP	IC 7474	2
2.	OR GATE	IC 7432	1
3.	IC TRAINER KIT	-	1
4.	CONNECTING WIRES	As per Required	

THEORY:

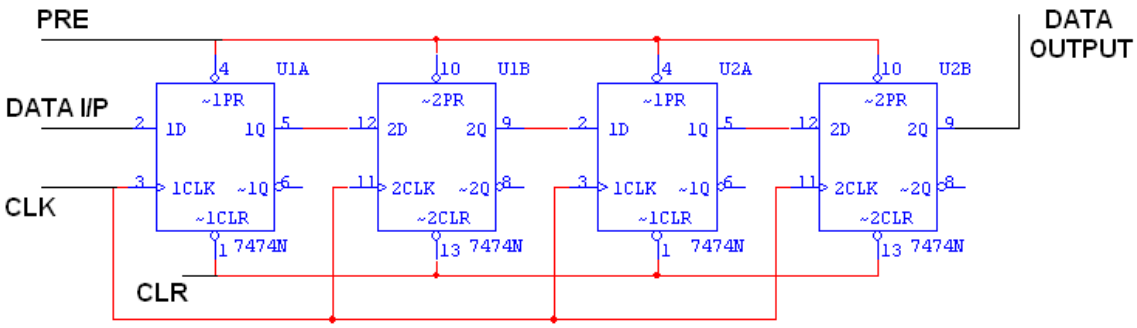
A register is capable of shifting its binary information in one or both directions is known as shift register. The logical configuration of shift register consist of a D-Flip flop cascaded with output of one flip flop connected to input of next flip flop. All flip flops receive common clock pulses which causes the shift in the output of the flip flop. The simplest possible shift register is one that uses only flip flop. The output 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:

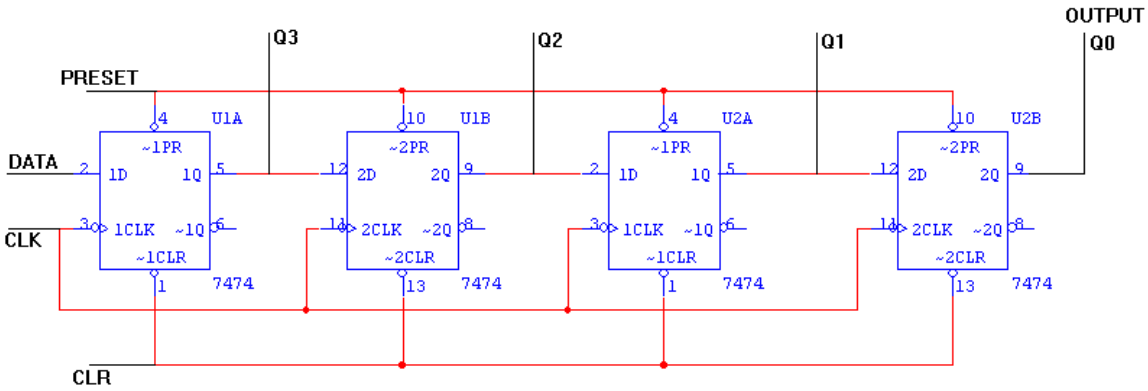


TRUTH TABLE:

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:

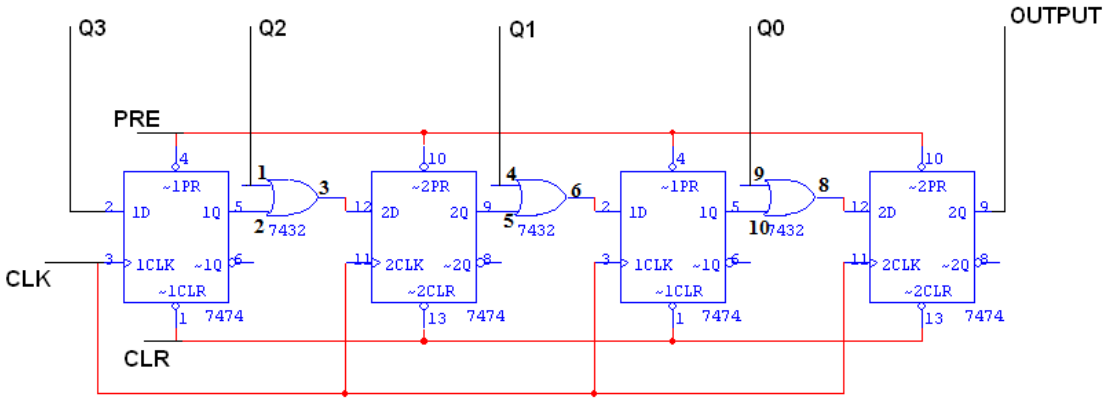


TRUTH TABLE:

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:

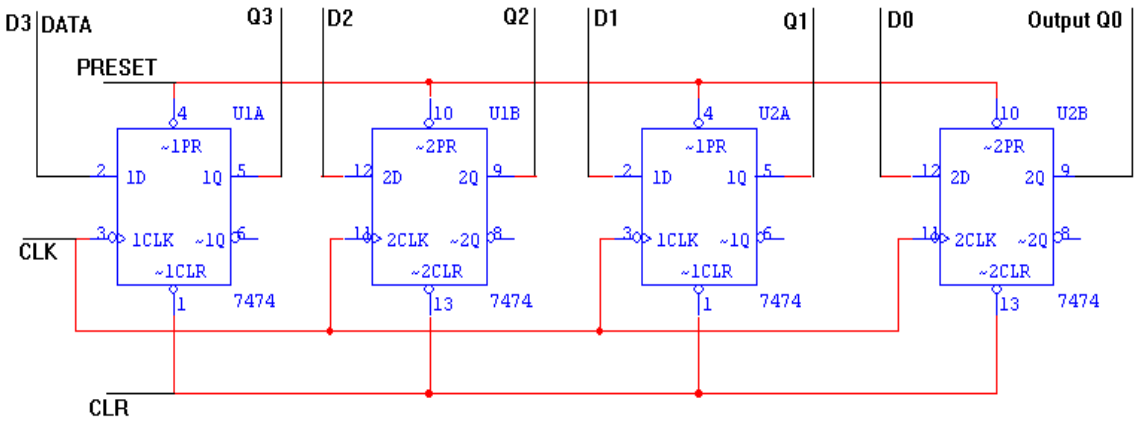


TRUTH TABLE:

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:

This SISO, SIPO, PISO and PIPO shift registers were designed and their truth tables were verified successfully.

SIMULATION OF COMBINATIONAL CIRCUITS USING HDL

Ex.No :
Date :

AIM:

To write the verilog code for half adder, full adder and Multiplexer using Xilinx software.

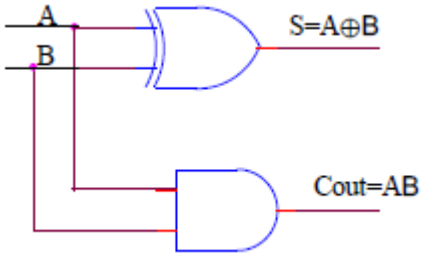
APPARATUS REQUIRED:

Personal Computer
Xilinx & ModelSim software

PROCEDURE:

- Draw the logic diagram of the corresponding system.
- Write the verilog code for that system.
- Enter the verilog code in Xilinx software.
- Check the syntax for error and simulate the code.
- Verify the output waveform.

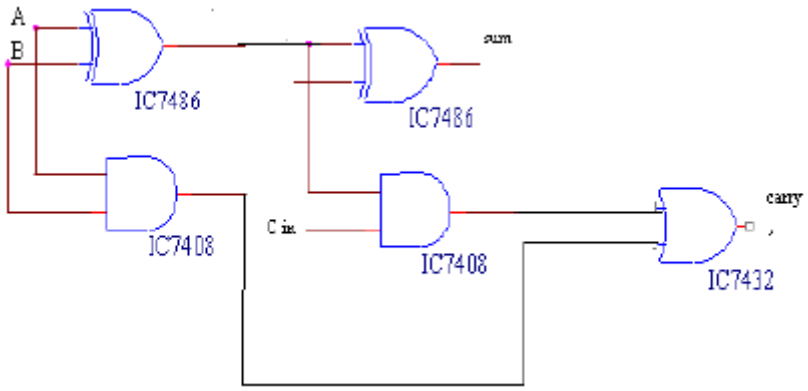
HALF ADDER



PROGRAM:

```
module halfadder(a, b, s, c);
input a;
input b;
output s;
output c;
xor(s,a,b);
and(c,a,b);
endmodule
```

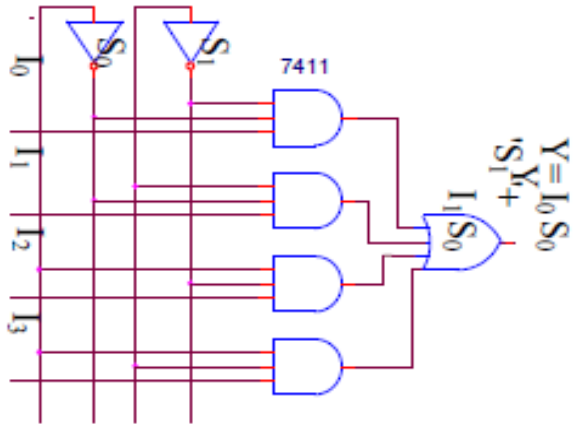
Full Adder:



PROGRAM:

```
module fulladder(a, b, c,  s, ca);
input a;
input b;
input c;
output s;
output ca;
xor(s,a,b);
and(c,a,b);
endmodule
wire si, c1, c2;
xor g1(s1,a,b);
and g2(c1,a,b);
xor g3(s,s1,c)
and g4(c2,s1,c);
or g5(ca, c1, c2)
endmodule
```

MULTIPLEXER:



Program:

```
module mux1(y, s0, s1, i0, i1, i2, i3);
output y;
input s0;
input s1;
input i0;
input i1;
inut i2;
input i3;
input i4;
always@(s0,s1);
begin
if(s0==1'b0&&s1==1'b0)
y=i0;
elseif(s0==1'b0&&s1==1'b1)
y=i1;
elseif(s0==1'b1&&s1==1'b0)
y=i2;
else
y=i3;
end
endmodule
```

RESULT:

Thus HDL program for combinational circuits were written using XiLinx software simulated successfully.

SIMULATION OF SEQUENTIAL CIRCUITS USING HDL

EX. NO :
DATE :

AIM:

To write the verilog code RS, D, JK Flip Flop and up counter using Xilinx software.

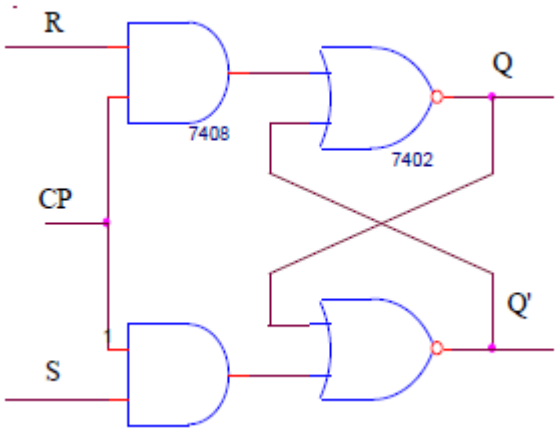
APPARATUS REQUIRED:

Personal Computer
Xilinx & ModelSim software

PROCEDURE:

- Draw the logic diagram of the corresponding system.
- Write the verilog code for that system.
- Enter the verilog code in Xilinx software.
- Check the syntax for error and simulate the code.
- Verify the output waveform

RS FLIP FLOP:
Circuit Diagram:

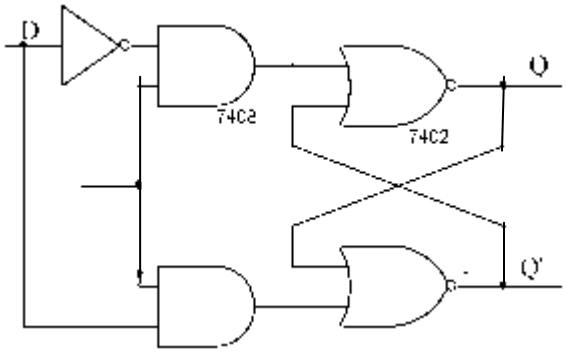


Program:

```
module sr1(clk, s, r, q);
input clk;
input s;
input r;
output reg q;
always@(posedge clk or s or r)
begin
if (clk=1)
if(s==1&&r==0)
q<=1;
else if (s==0 && r==0)
q<=q;
else
q<=0;
end
end module
```

D FLIP FLOP:

Circuit diagram:

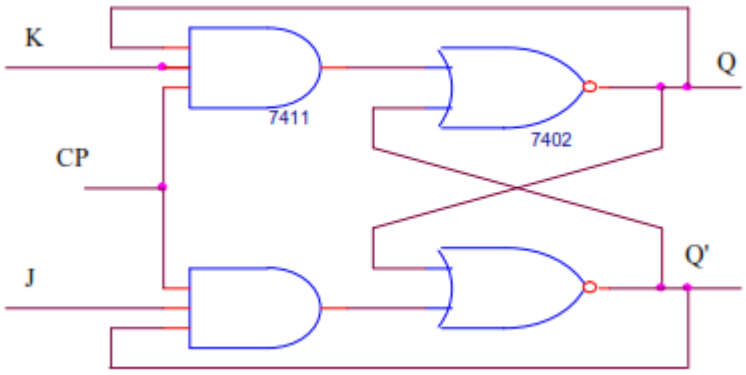


Program:

```
module dff1 (d, clk, rst, q);
input d;
input clk;
input rst;
output q;
reg q;
always@(posedge clk)
if (rst=0)
q<=0;
else
q<=1;
end module
```

JK FLIP FLOP:

Circuit diagram:



Program:

```
module sr1(clk, j,k, q);
input clk;
input j;
input k;
output reg q;
always@(posedge clk or j or k)
begin
if (clk=1)
if(j==1&&k==0)
q<=1;
else if (j==0 && k==0)
q<=q;
else
q<=~0;
end
end module
```

UP COUNTER:

Program:

```
module upcounter(clk, N, a);
input clk;
input [3:0]a;
initial a=4'b0000;
always@(negedge clk)
a=(a==N)?4'b0000:a+1'b1;
endmodule
```

RESULT:

Thus HDL program for combinational circuits were written using XiLinx software and simulated successfully.