SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM NAGAR, KATTANKULATHUR – 603 203.

Department Of Electronics and Communication Engineering

LABORATORY MANUAL

EC3366 - DIGITAL SYSTEMS DESIGN LABORATORY

Regulation - 2023

Semester/Branch	**:**	**III semester ECE**
Academic Year	**:**	**2025 -26 (ODD)**
Course Coordinator	**:**	**Dr. K. Lekha, A.P (O.G)**
Prepared By	**:**	**Dr. C. Amali, Associate Professor** **Dr. C. Saravanakumar, A.P (Sel.G)** **Dr. N. Jothy, A.P (Sr.G)** **Mr. S. Senthilmurugan, A.P (Sel.G)**

SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur – 603 203

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

VISION OF THE INSTITUTE

Educate to excel in social transformation

MISSION OF THE INSTITUTE

- To contribute to the development of human resources in the form of professional engineers and managers of international excellence and competence with high motivation and dynamism, who besides serving as ideal citizen of our country will contribute substantially to the economic development and advancement in their chosen areas of specialization.
- To build the institution with international repute in education in several areas at several levels with specific emphasis to promote higher education and research through strong institute-industry interaction and consultancy.

VISION OF THE DEPARTMENT

To excel in the field of electronics and communication engineering and to develop highly competent technocrats with global intellectual qualities.

MISSION OF THE DEPARTMENT

M1: To train the students with the state-of-art technologies and to develop innovative solutions to cater to the societal needs

M2: To orient the students towards global career with universal moral values and professional ethics

PROGRAM OUTCOMES

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOME(PSOs)

PSO1: Ability to apply the acquired knowledge of basic skills, mathematical foundations, principles of electronics, modelling and design of electronics based systems in solving engineering Problems.

PSO2: Ability to understand and analyze the interdisciplinary problems for developing innovative sustained solutions with environmental concerns.

SYLLABUS

EC3366 - DIGITAL SYSTEMS DESIGN LABORATORY L T P C 3 0 2 4

OBJECTIVES:

1.To understand the Digital fundamentals, Boolean algebra and its applications in digital systems.

2. To Practice with the design of combinational digital circuits using logic gates.

3. To bring out the analysis and design procedures for synchronous sequential circuits.

4. To explore the design procedures of an asynchronous sequential circuits.

5. To introduce semiconductor memories and related technology.

6. To design combinational and sequential circuits using HDL.

LIST OF DIGITAL EXPERIMENTS

1. Design of adders and subtractors & code converters.
2. Design of Multiplexers & Demultiplexers.
3. Design of Encoders and Decoders.
4. Design of Magnitude Comparators.
5. Design and implementation of counters using flip-flops.
6. Design and implementation of shift registers.

TOTAL: 30 PERIODS

OUTCOMES:

On completion of the course, the student will be able to

1. Use Boolean algebra and simplification procedures relevant to digital logic.
2. Apply the logical knowledge in the design of combinational circuits.
3. Analyze the concepts of sequential circuits and design sequential circuits in terms of state machines.
4. Design and analyze asynchronous sequential circuits.
5. Build logic gates and use programmable devices.
6. Implement combinational and sequential circuits using HDL.

COURSE OUTCOMES - PROGRAM OUTCOMES MATRIX

CO	PO												PSO	
	1	2	3	4	5	6	7	8	9	10	11	12	1	2
EC3366.1	3	2	2	-	-	-	-	-	3	-	-	3	3	2
EC3366.2	2	3	3	2	2	-	-	-	3	-	-	1	2	2
EC3366.3	2	3	3	3	2	-	-	-	3	-	-	2	2	2
EC3366.4	2	3	3	3	2	-	-	-	3	-	-	2	2	2
EC3366.5	2	2	1	1	2	-	-	-	3	-	-	2	2	2
EC3366.6	2	3	3	2	3	-	-	-	3	-	-	2	2	2
AVG	2	3	3	2	2	-	-	-	3	-	-	2	2	2

LAB Requirements for a Batch of 30 students (3 students per experiment):

S.NO	EQUIPMENTS FOR DIGITAL LAB	REQUIRED
1.	Dual power supply/ single mode powersupply.	30 Nos
2.	IC Trainer Kit.	30 Nos
3.	Bread Boards	30 Nos
4.	Seven segment display	30 Nos
5.	Multimeter	10 Nos
6.	ICs each 50 Nos. 7400/ 7402 / 7404 / 7486 / 7408 / 7432 / 7483 / 74150 / 74151 / 74147 / 7445 / 7476/7491/ 555 / 7494 / 7447 / 74180 /7485 / 7473 / 74138 / 7411 / 7474	–

LIST OF EXPERIMENTS

CYCLE-I

Expt. No.	Name of the Experiment	Page No.
1	Design of adders and subtractors & code converters.	
2	Design of Multiplexers & Demultiplexers.	
3	Design of Encoders and Decoders.	
4	Design of Magnitude Comparators.	
5	Design and implementation of counters using flip-flops.	
6	Design and implementation of shift registers.	

TOPIC BEYOND SYLABBUS

Expt. No.	Name of the Experiment	Page No.
7	Design of Half adder and Full adder using the VHDL code	
8	Design of D flip flop using the VHDL code	

EXPT NO. : 0 **DATE :**

STUDY OF LOGIC GATES AND FLIP FLOPS

AIM:

To study the basic logic gates, Flip Flop's and verifies their truth tables.

COMPONENTS REQUIRED:

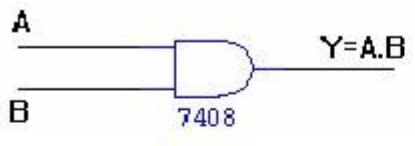
SL No.	COMPONENT	SPECIFICATION	QTY
1.	AND GATE	IC 7408	1
2.	OR GATE	IC 7432	1
3.	NOT GATE	IC 7404	1
4.	NAND GATE 2 I/P	IC 7400	1
5.	NOR GATE	IC 7402	1
6.	X-OR GATE	IC 7486	1
7.	X-NOR GATE	IC 74266	1
8.	AND GATE 3 I/P	IC 7411	1
9.	NAND GATE 3 I/P	IC 7410	1
10.	D –Flip Flop	IC 7474	1
11.	JK – Flip Flop	IC 7476	1
12.	IC TRAINER KIT	-	1
13.	PATCH CORD	-	As required

PRE-LAB EXERCISE:

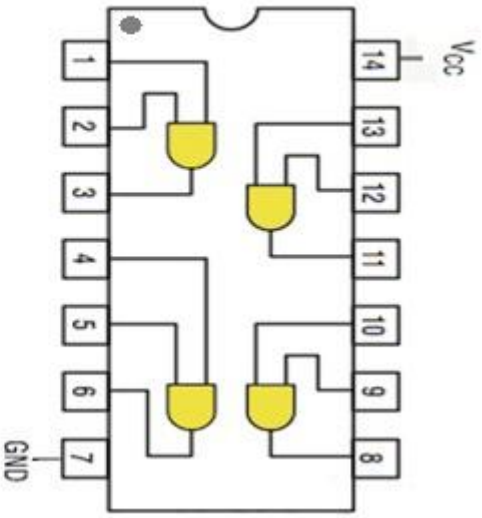
1. Differentiate Bit, Byte, and Nibble.
2. Explain the term universal gate.
3. What is a truth table? What is its significance in logic operations?
4. Define Minterm.& Maxterm.
5. What is sequential logic circuit?
6. What is latch?
7. What are the types of Latch?
8. What is Flip-flop?
9. What is the difference between latch & Flip-Flop?
10. What is the need of triggering?
11. What are the types of triggering?
12. What are the types Flip-flop?

AND GATE:

SYMBOL



PIN DIAGRAM

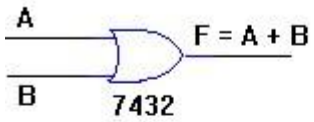


TRUTH TABLE

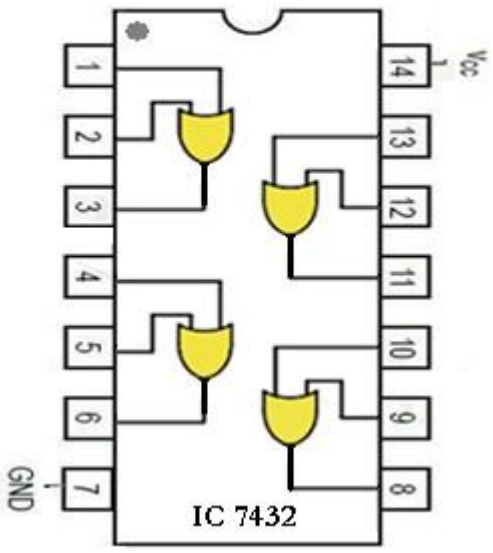
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

OR GATE:

SYMBOL :



PIN DIAGRAM :



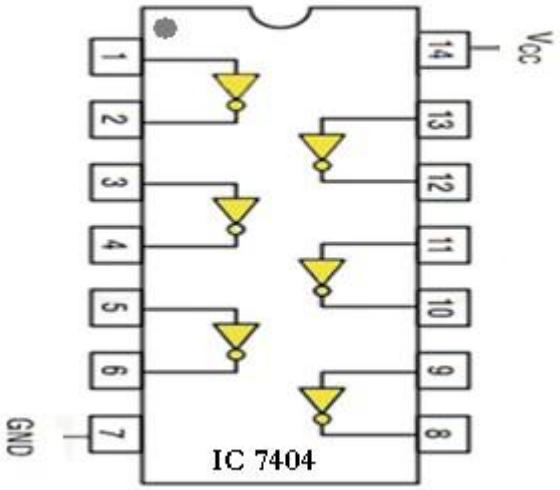
TRUTH TABLE

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

NOT GATE:

SYMBOL

PIN DIAGRAM

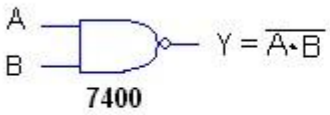


TRUTH TABLE :

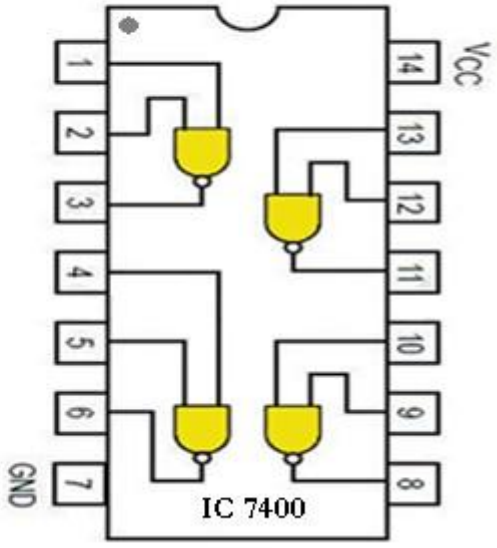
A	$\overline{A}$
0	1
1	0

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

THEORY:

Digital electronics is based on the binary number system. Instead of voltages which vary continuously, as in analog electronics, digital circuits involve voltages which take one of only two possible values. In our case these are 0 and 5 volts (TTL logic), but they are often referred to as LOW and HIGH, or FALSE and TRUE, or as the binary digits 0 and 1.The basic building blocks of digital electronics are logic gates which perform simple binary logic functions (AND, OR, NOT, etc.). From these devices, one can construct more complex circuits to do arithmetic, act as memory elements, and so on. Logic gates and other digital components come in the form of integrated circuits (ICs) which consist of small semiconductor chips packaged in a ceramic or plastic case with many pins. The ICs are labeled by numbers like 74LSxx, where LS is technologies and xx is a number identifying the type of device.

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 (IC 7408):

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 (IC 7432):

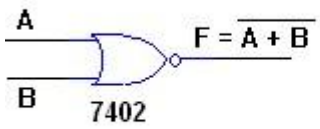
The OR gate performs a logical addition commonly known as OR function. The output is high when any one of the inputs is high. The output is low level when both the inputs are low.

NOT GATE (IC 7404):

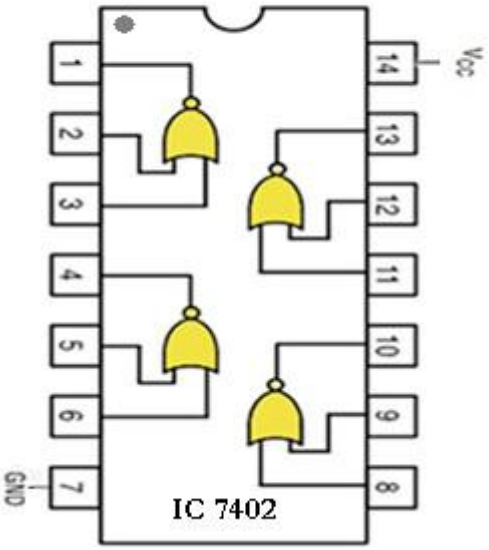
The NOT gate is called an inverter. The output is high when the input is low. The output is low when the input is high. (i.e., output is Complement of the input)

NOR GATE:

SYMBOL :



PIN DIAGRAM :

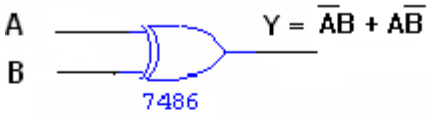


TRUTH TABLE

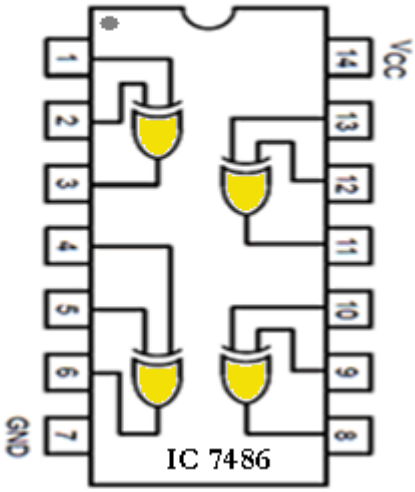
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

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

NAND GATE (IC 7400):

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 (IC 7402):

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 (Exclusive OR) GATE (IC 7486):

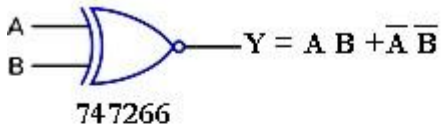
The output is high when any one of the inputs is high. The output is low when both the inputs are low and both the inputs are high.

X-NOR (Exclusive NOR) GATE (IC 74266):

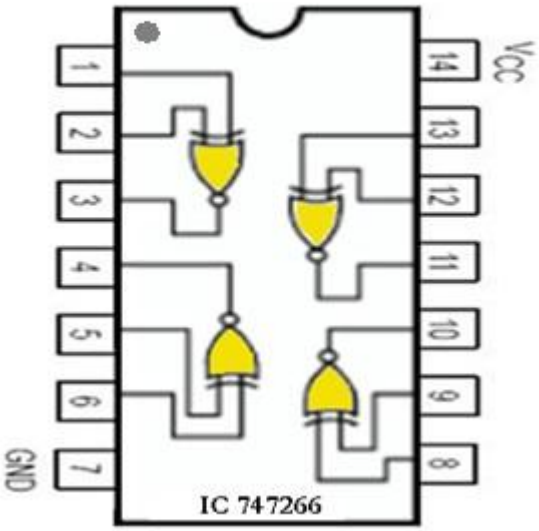
The output is high when both the inputs are low and both the inputs are high. The different inputs will produce low output (i.e. Logic ‘0’).

X-NOR GATE :

SYMBOL



PIN DIAGRAM

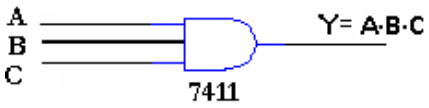


TRUTH TABLE :

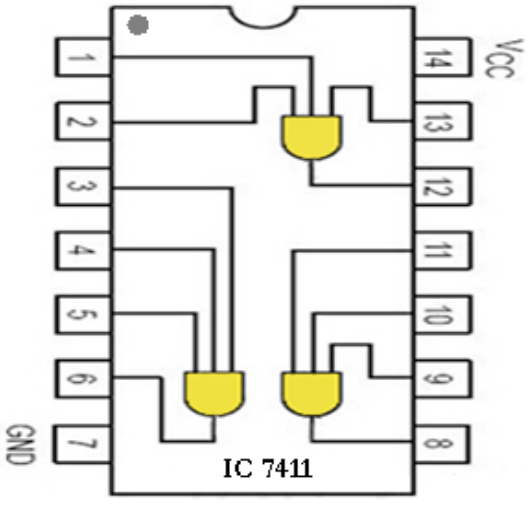
A	B	AB + $\overline{A}\overline{B}$
0	0	1
0	1	0
1	0	0
1	1	1

3-INPUT AND GATE :

SYMBOL



PIN DIAGRAM



TRUTH TABLE

A	B	C	A·B·C
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

PROCEDURE:

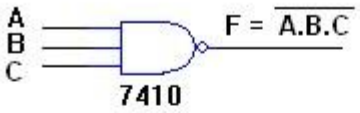
(i) Connections are given as per circuit diagram.

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

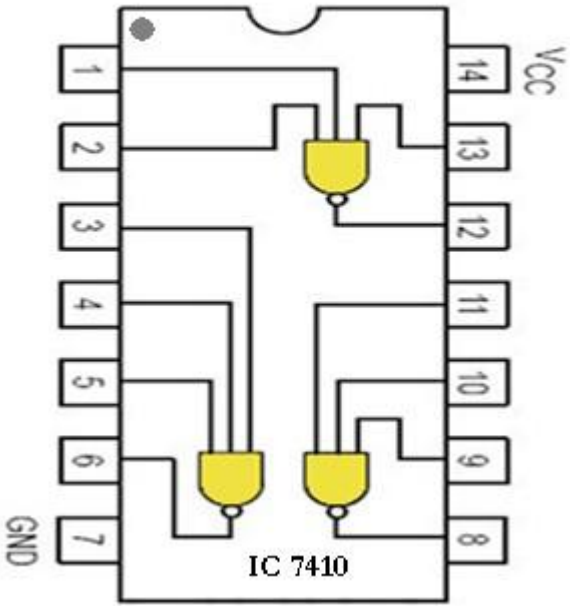
(iii) Observe the output and verify the truth table.

3-INPUT NAND GATE:

SYMBOL



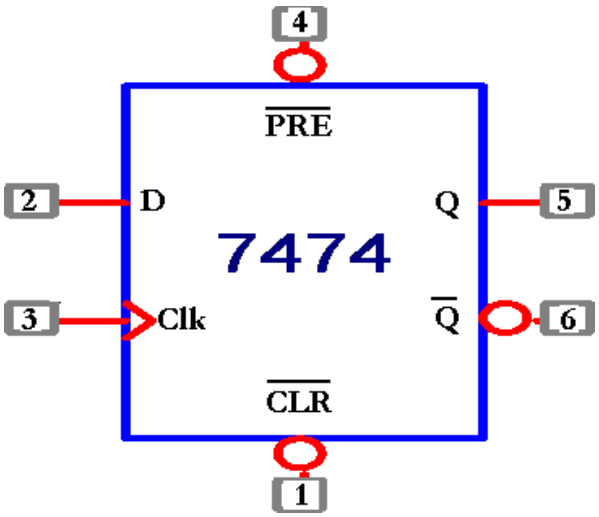
PIN DIAGRAM



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

LOGIC DIAGRAM:



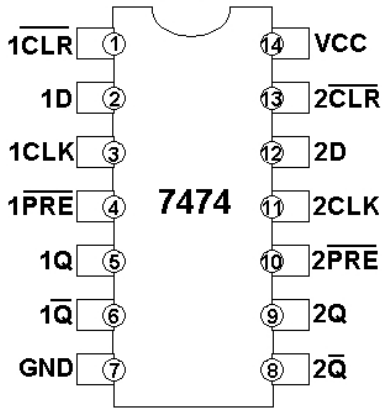
TRUTH TABLE:

Clock	Input D	Output Q
L	X	No change
H	0	0
H	1	1

FUNCTION TABLE:

CLK	D	Q	Q'
1	1	1	0
1	0	0	1

PIN DIAGRAM:



Theory:

A flip-flop is a simplest kind of sequential circuit that has only two states. It can be a binary 1 or 0. It is also used as a memory cell as it stores one bit of information.

D FLIPFLOP:

It is a single input version of the flip-flop. It accepts single data and provides output which is same as the applied input. It is called Delay flip flop. The output of the D FF as follows the input.

JK FLIP-FLOP:

JK flip-flops are far more versatile than RS and D-type flip-flops and they can perform manymore functions than simple latches. JK Flip-flops 7476 which use level-clocking. It is important to remember that for normal operation the preset and clear inputs should be held (tiedpositive).

PROCEDURE:

i. The power supply to D flip-flop and JK flip-flop is switched on.

ii. The truth tables of flip flops are verified. In a JK flip flop, first we will consider the asynchronous operation of the flip flops. The J and K inputs have no effects on the asynchronous operation.

APPLICATIONS:

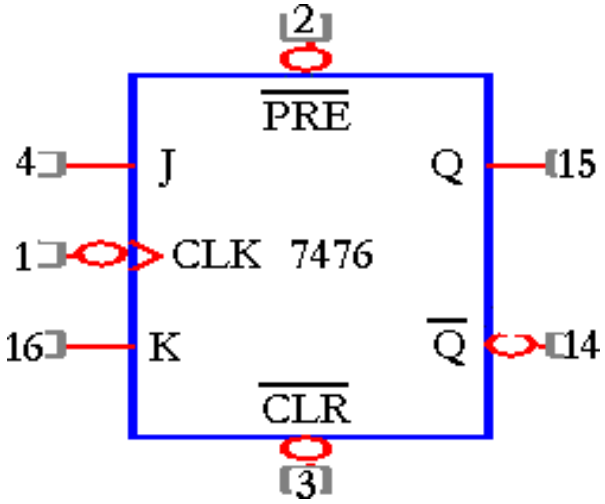
- It finds immense use in parallel data storage.
- It is used in shift registers and counter circuits.
- It can also be used in frequency divider.

POST-LAB EXERCISE:

1. What is meant by logic family?
2. State two advantages of CMOS logic.
3. What is a tri-state gate?
4. Define fan out for a logic circuit.
5. What are the applications of latch?
6. What is Excitation table?
7. What is Master-slave JK Flip-flop?
8. What is called transition (state) table?
9. Draw the symbol of SR, JK, T & D Flip-flop?
10. How to realize one Flip-flop using another Flip-flop?
11. What are the applications of Flip-flop?
12. What is the full form of SR, JK, T & D?

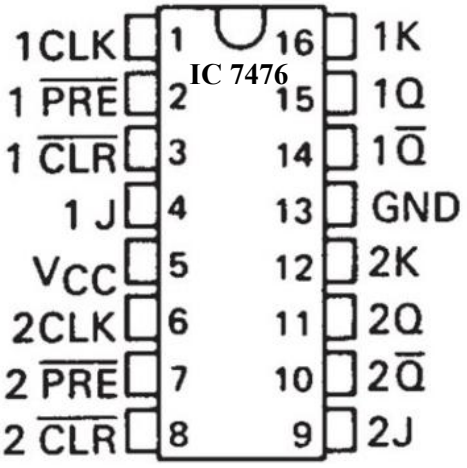
LOGIC DIAGRAM:

Truth Table



Inputs			Output Q
CLK	J	K	
X	X	X	Invalid
X	X	X	1
X	X	X	0
clk	0	0	Nochange
clk	0	1	0
clk	1	0	1
clk	1	1	Toggle

PIN DIAGRAM:



RESULT:

Thus different types of logic gates and Flip flops are studied and their truth table were verified.

EXPT NO.:1a **DATE :**

HALF & FULL ADDERS AND SUBTRACTORS

AIM:

To study by design and construct half adder, full adder, half subtractor and full subtractor circuits and verify their truth tables using logic gates.

APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	Quad 2 input AND gate	IC 7408	1
2.	Quad 2 input X-OR gate	IC 7486	1
3.	Hex 1 input NOT gate	IC 7404	1
4.	Quad 2 input OR gate	IC 7432	1
3.	IC Trainer kit	-	1
4.	Patch Cords	-	As required

THEORY:

HALF ADDER:

A half adder is a combinational circuit needs two binary inputs and two binary outputs. The input variables designate the augend and addend bits, the output variables produce the sum and carry. The half-adder can be implemented with an exclusive –OR and an 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. In full adder sum output will be taken from X-OR Gate, carry output will be taken from OR Gate.

HALF SUBTRACTOR:

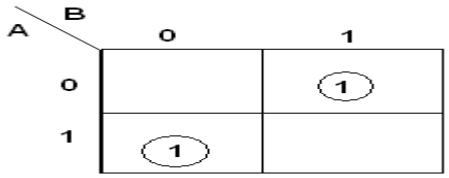
The half subtractor is constructed using X-OR and AND Gate. The half subtractor has two input and two outputs. The outputs are difference and borrow. The difference can be applied using X-OR Gate, borrow output can be implemented using an AND Gate and an inverter.

HALF ADDER

TRUTH TABLE:

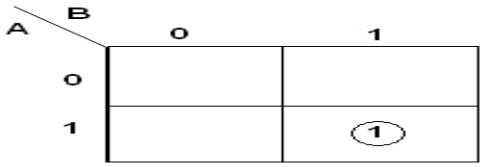
Inputs		Outputs	
A	B	CARRY	SUM
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

K-Map for SUM:



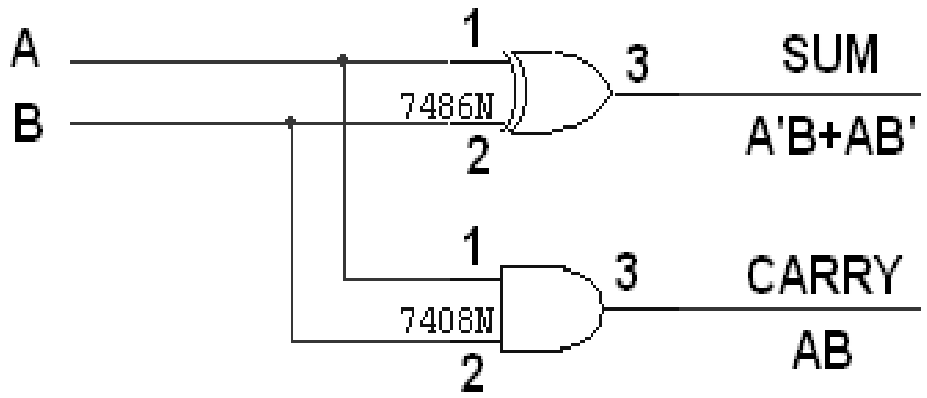
SUM = A'B + AB'
= $A \oplus B$

K-Map for CARRY:



CARRY = AB

LOGIC DIAGRAM:



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.

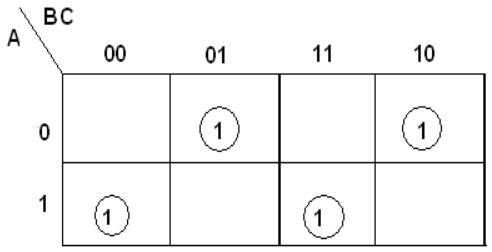
FULL ADDER

FULL ADDER USING TWO HALF ADDER:

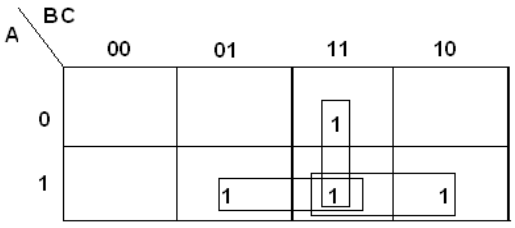
TRUTH TABLE:

Inputs			Outputs	
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:



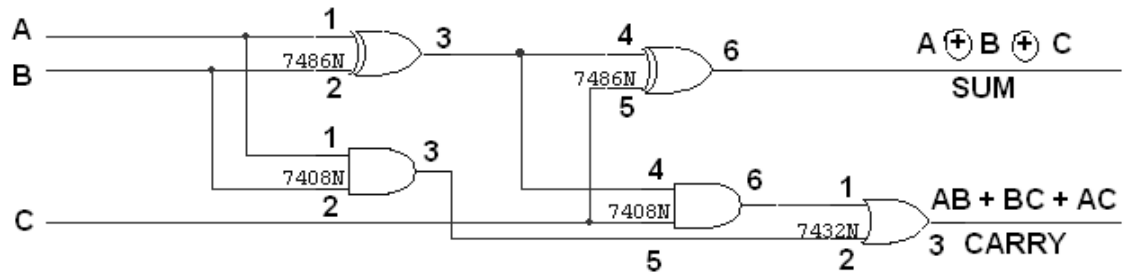
K-Map for CARRY:



SUM = A'B'C + A'BC' + ABC' + ABC
= C(A'B'+AB) + C'(A'B+AB')
= A ⊕ B ⊕ C

CARRY = AB + BC + AC

LOGIC DIAGRAM:



PROCEDURE:

(i) Connections are given as per circuit diagram.

(ii) Switch ON the digital lab trainer kit

(iii)Inputs are given using the toggle switch

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

VIVA QUESTIONS:

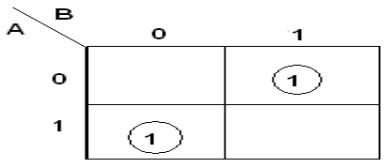
1. What is an adder?
2. What are the types of adders?
3. What is a subtractor?
4. What are the types of subtractors?
5. Explain the working of a half adder.
6. What are the limitations of a half adder?
7. How does a full adder work?
8. Why do we use a full adder instead of a half adder in multi-bit addition?
9. Can you derive the Boolean expressions for the sum and carry outputs of a full adder?
10. How does a carry-lookahead adder reduce propagation delay?

HALF SUBTRACTOR

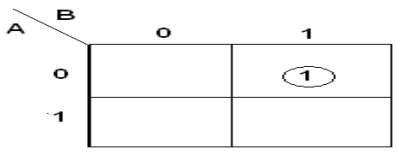
TRUTH TABLE:

Inputs		Outputs	
A	**B**	**BORROW**	**DIFFERENCE**
0	**0**	**0**	**0**
0	**1**	**1**	**1**
1	**0**	**0**	**1**
1	**1**	**0**	**0**

K-Map for DIFFERENCE:



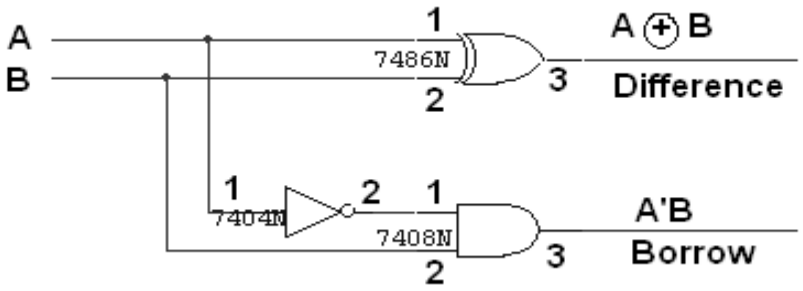
K-Map for BORROW:



DIFFERENCE = A'B + AB'
= $A \oplus B$

BORROW = A'B

LOGIC DIAGRAM:



FULL SUBTRACTOR

TRUTH TABLE:

Inputs			Outputs	
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**

RESULT:

Thus adder and subtractor are constructed and their truth tables are verified using logic gates

EXPT NO.:1b **DATE :**

DESIGN AND IMPLEMENTATION OF CODE CONVERTORS USING LOGIC GATES

AIM:

To design and implement 4-bit

(i) BCD to excess-3 code converter

(ii) Excess-3 to BCD code converter

(iii) Binary to gray code converter

(iv) Gray to binary code converter

COMPONENTS 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 required

PRE-LAB EXERCISE:

1. What is the need for code converter?
2. How do you convert binary numbers to corresponding Gray codes using a converter?
3. Define a code converter logic circuit.
4. How do you convert Gray code numbers to corresponding binary numbers using a converter?
5. What are the steps involved in the BCD to binary conversion process?

THEORY:

A code converter is a combinational logic circuit that changes the data presented in one type of binary code to another type of binary code. To convert from BCD 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. The Excess-3 code represents a decimal number, in binary form, as a number greater than 3. An excess-3 code is obtained by adding 3 to a decimal number. The excess-3 code is a self -complementing code.

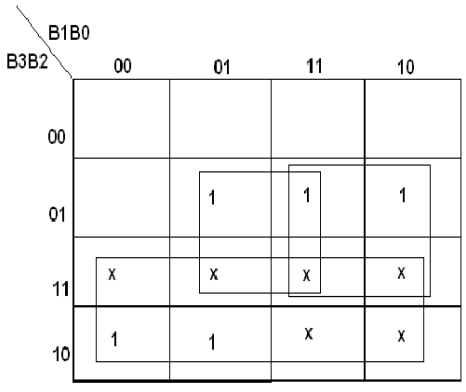
Each one of the four maps represent one of the four outputs of the circuit as a function of the four input variables. A two-level logic diagram may be obtained directly from the Boolean expressions derived by the maps. These are various other possibilities for a logic diagram that implements this circuit.

BCD TO EXCESS-3 CONVERTOR

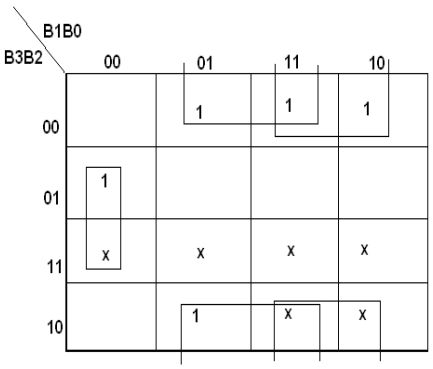
TRUTH TABLE:

BCD input				Excess – 3 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 and E_2:

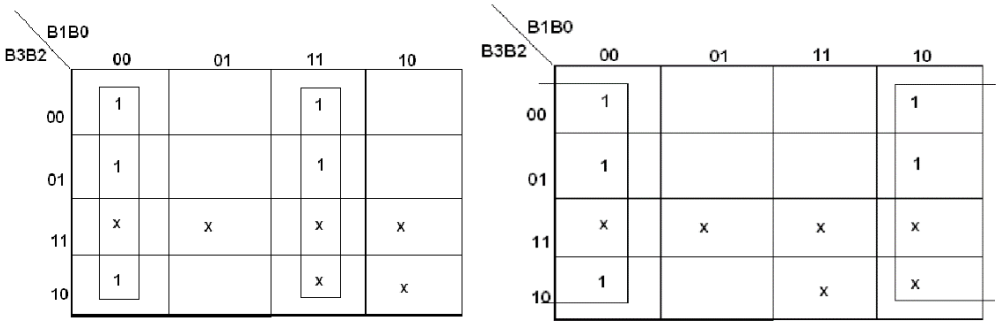


E3 = B3 + B2 (B0 + B1)



E2 = B2 ⊕ (B1 + B0)

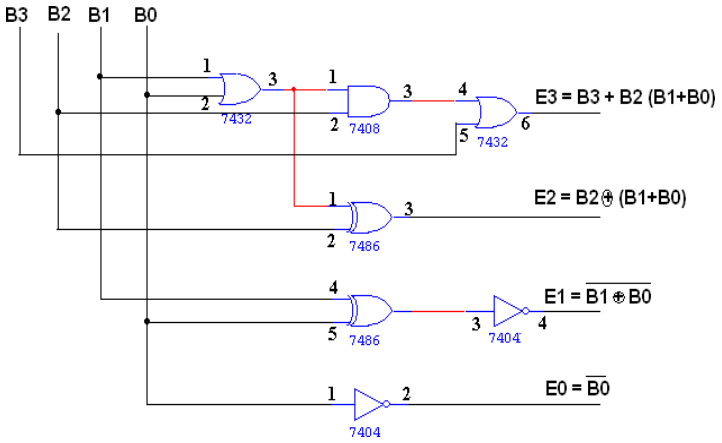
K-Map for E_1 and E_0:



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

$E0 = \overline{B0}$

LOGIC DIAGRAM:

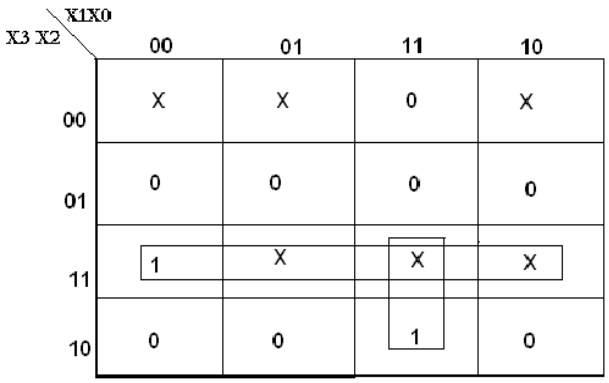


EXCESS-3 TO BCD CONVERTOR

TRUTH TABLE:

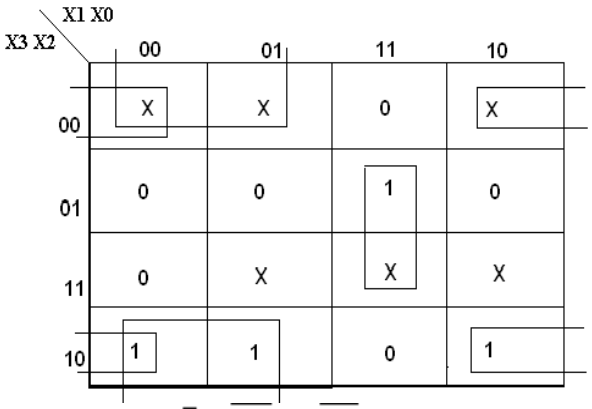
Excess – 3 Input				BCD Output			
X3	X2	X1	X0	D	C	B	A
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1

K-Map for A:



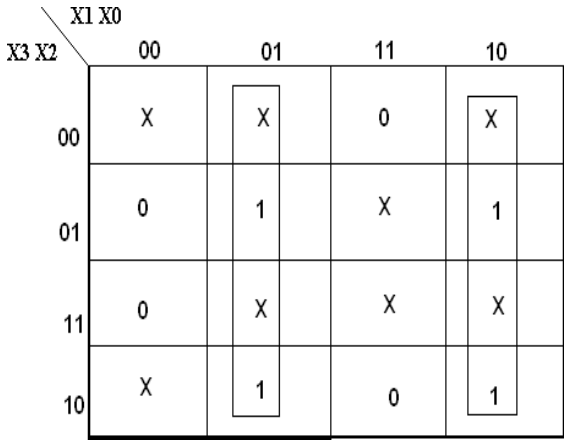
A = X3 X2 + X1 X0 X3

K-Map for B:



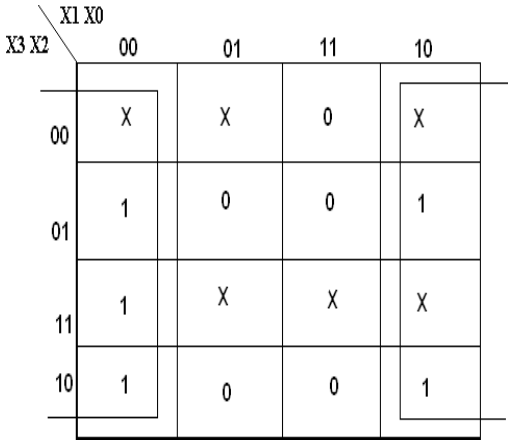
$B = X2 \oplus (\overline{X_1} + \overline{X_0})$

K-Map for C:



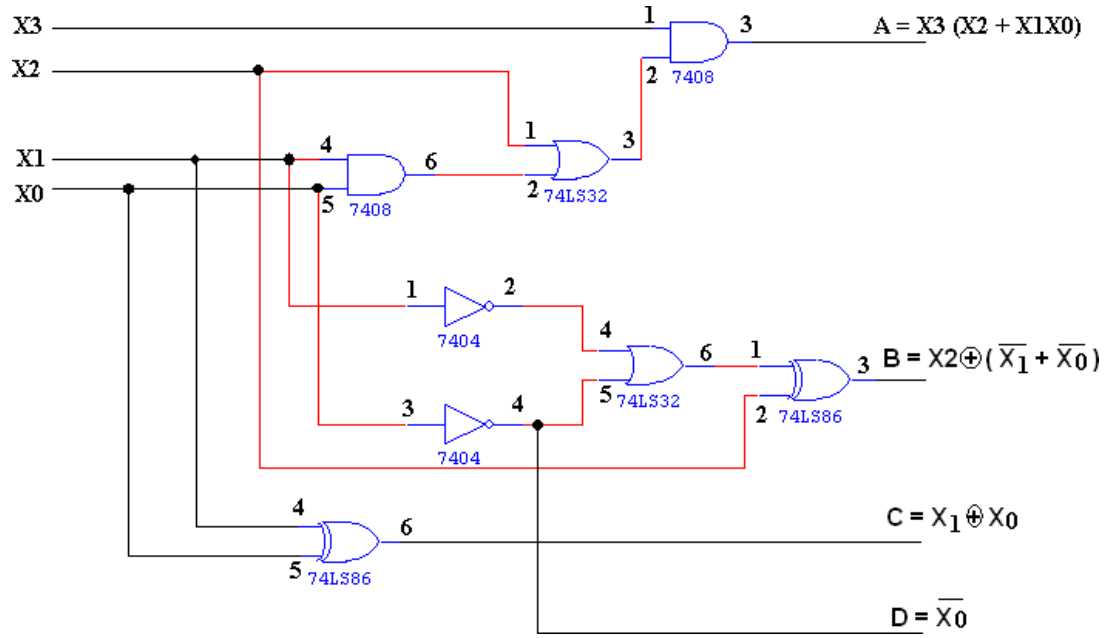
$C = X_1 \oplus X_0$

K-Map for D:



$D = \overline{X_0}$

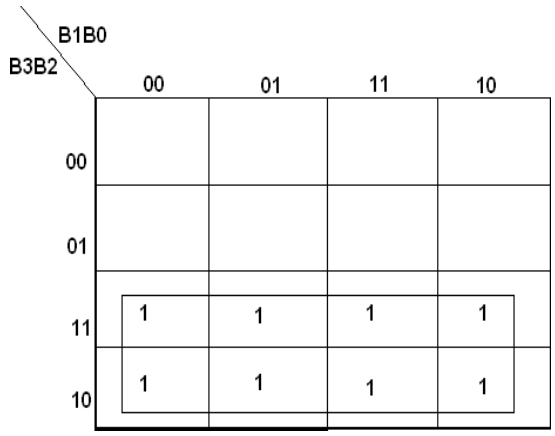
LOGIC DIAGRAM:



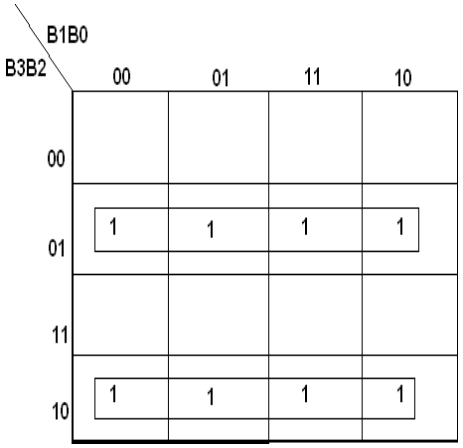
TRUTH TABLE:

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 and G2



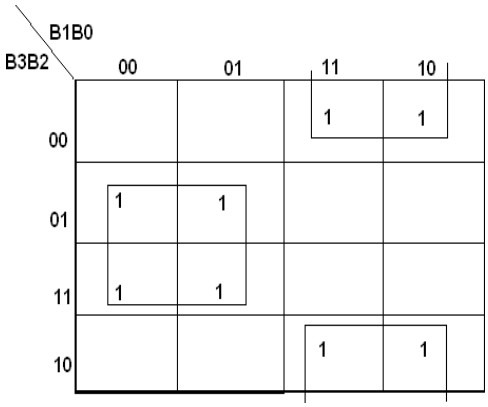
$G_3 = B_3$



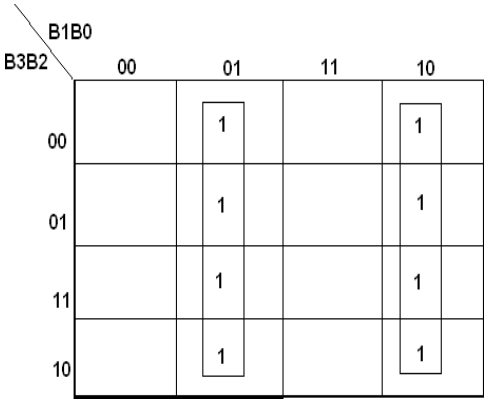
$= B'_3 B_2 + B'_2 B_3$

$G2 = B3 \oplus B2$

K-Map for G_1 and G_0:



$G1 = B1 \oplus B2 = B_1 B'_2 + B_2 B'_1$



$G0 = B1 \oplus B0 = B_1 B'_0 + B'_1 B_0$

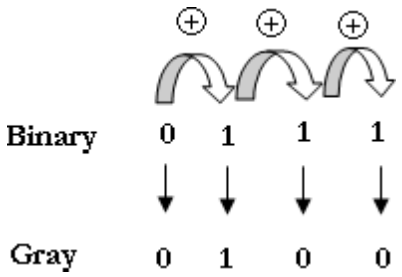
THEORY:

A code converter is a combinational logic circuit that changes the data presented in one type of binary code to another type of binary code.

BINARY TO GRAY CODE CONVERSION:

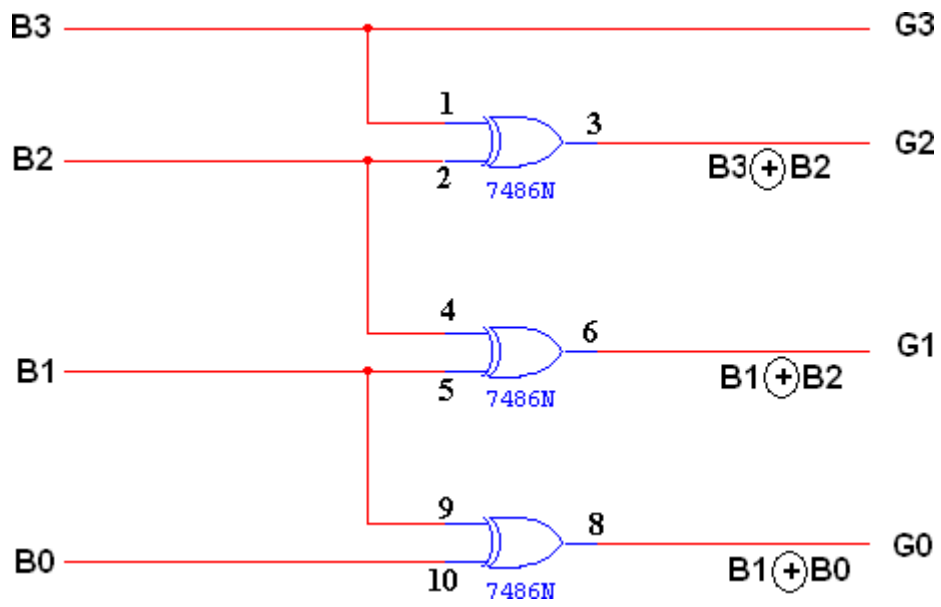
The advantage of gray code is that two adjacent code numbers differ by only one bit. It is used in some types of analog to digital converters.

- The MSB of the gray code is same as the MSB of the binary code.
- The second bit of the gray code is obtained by adding the first & second bits of binary code after eliminating carry, (or) XOR operation of two inputs.
- The third bit of the gray code is obtained by adding the second & third bits of binary code after eliminating carry & so on.



Gray code is a non-weighted code

LOGIC DIAGRAM:

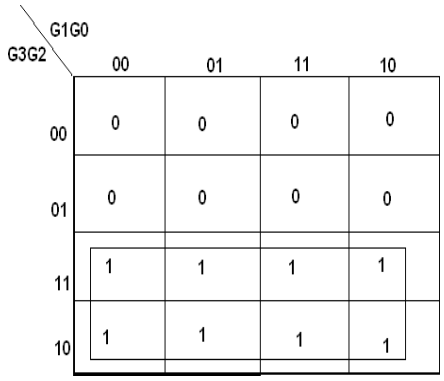


GRAY CODE TO BINARY CONVERTOR

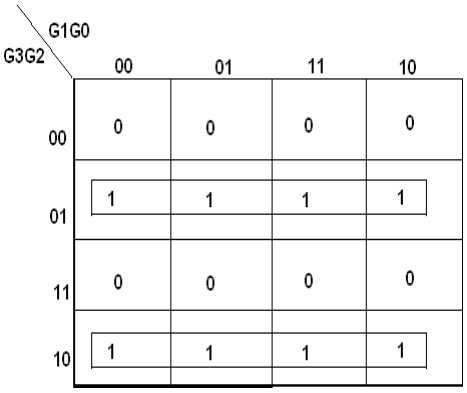
TRUTH TABLE:

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 and B_2:



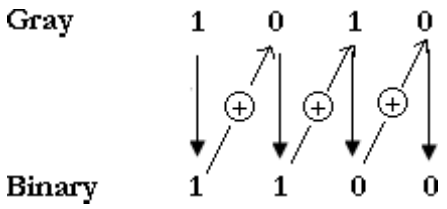
B3 = G3



B2 = G3⊕G2

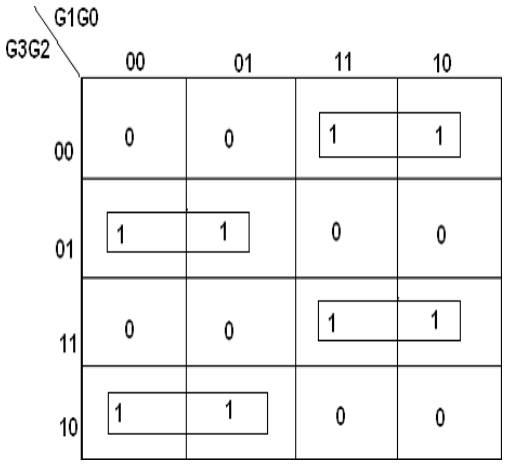
GRAY TO BINARY CODE CONVERSION:

- The MSB of the binary code is the same as the MSB of the gray code.
- The second bit of the binary code is obtained by adding the second bit of gray code and first bit of the binary code after eliminating carry.
- The third bit of the binary code is obtained by adding the second bit of binary code & third bit of gray code after eliminating carry & so on.

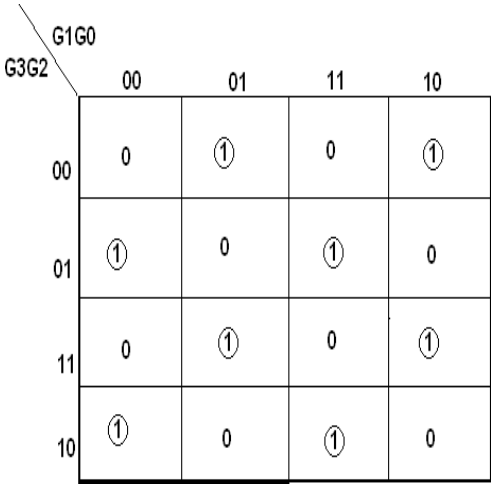


The input variable are designated as B3, B2, B1, B0 and the output variables are designated as G3, G2, G1, Go. from the truth table, combinational circuit is designed. The Boolean functions are obtained from K-Map for each output variable. A code converter is a circuit that makes the two systems compatible even though each uses a different binary code.

K-Map for B_1 and B_0:

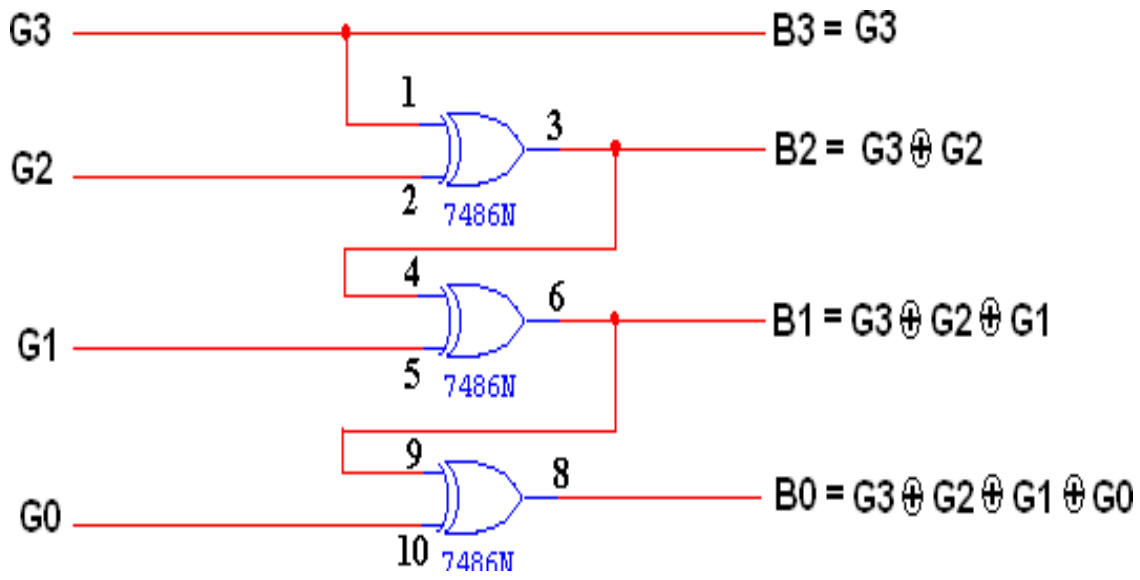


$B1 = G3 \oplus G2 \oplus G1$



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

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

APPLICATIONS:

Code converters are circuits that make two systems compatible even though each of them used a different mode.

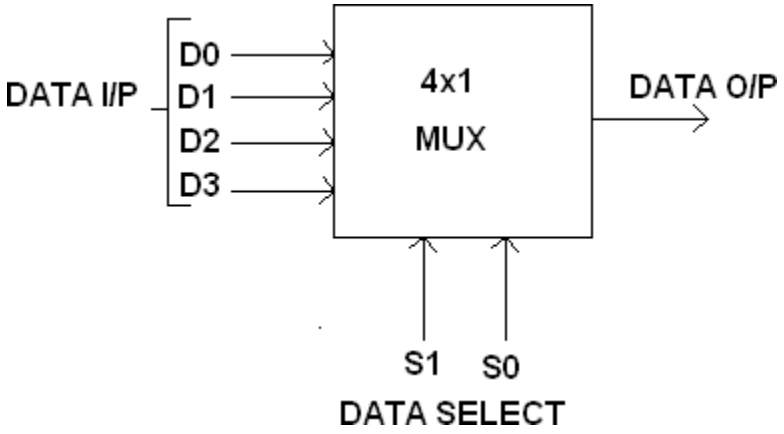
POST-LAB EXERCISE

1. What is Gray code?
2. What are alphanumeric codes?
3. Draw the flow diagram for Gray to binary conversion.
4. What is ASCII code?
5. What is EBCDIC?

RESULT: Thus the different types of code converters were designed, constructed and their truth tables were verified.

4:1 MULTIPLEXER

BLOCK DIAGRAM:



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

EXPT NO.:2 **DATE :**

DESIGN AND IMPLEMENTATION OF MULTIPLEXER AND DEMULTIPLEXER USING LOGIC GATES

AIM:

i. To design and implement 4 x 1 MUX and 1x 4 DEMUX using logic gates and study of IC 74150 and IC 74154.

COMPONENTS 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 required

PRE-LAB EXERCISE:

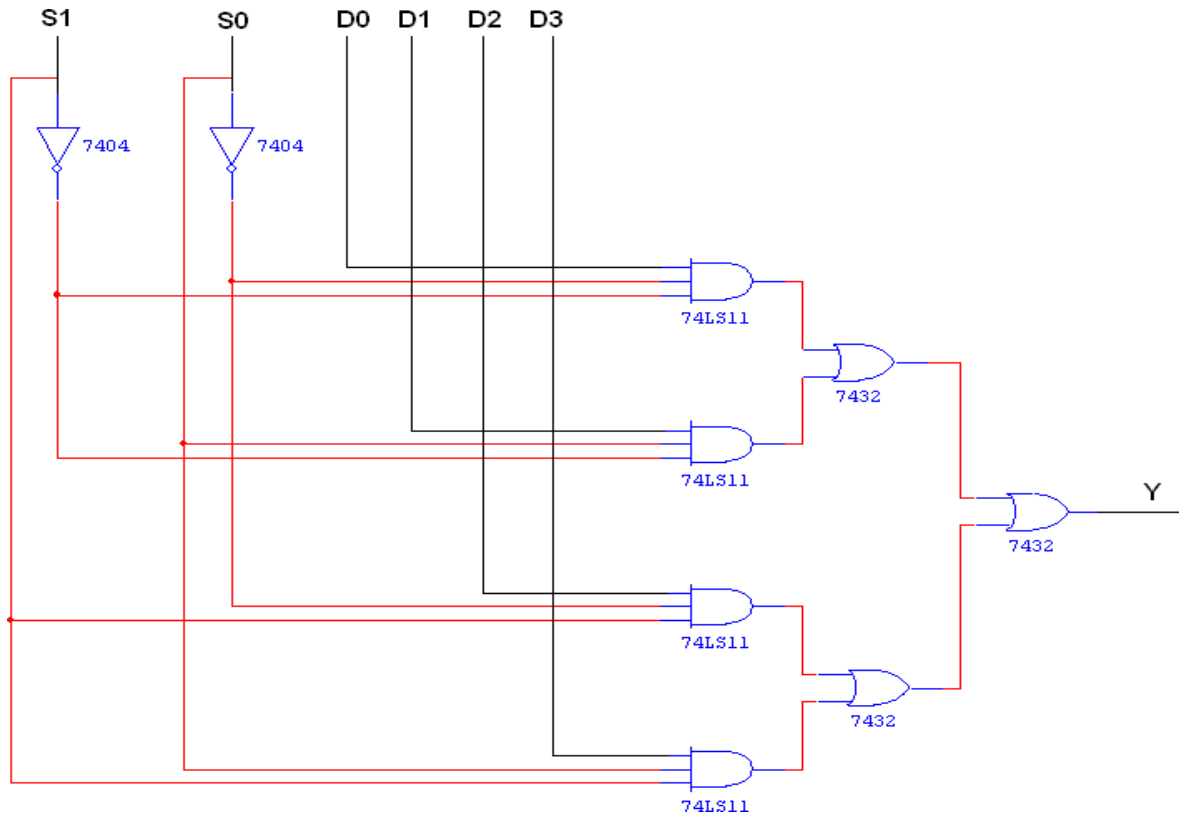
1. What is another name of multiplexer?
2. What is a four channel multiplexer?
3. What is the function of a multiplexer's select inputs?
4. What are the major applications of multiplexers?
5. Identify each MSI device? (a)74157 (b) 71151 (c) 74150

THEORY:

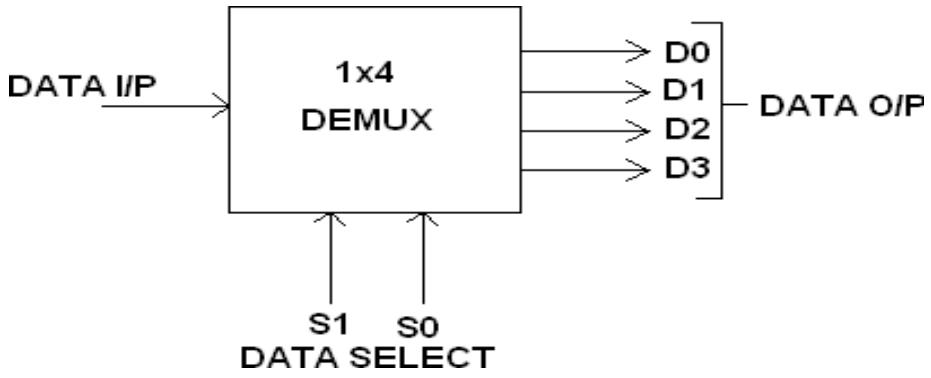
MULTIPLEXER:

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.

LOGIC DIAGRAM FOR MULTIPLEXER:



BLOCK DIAGRAM:



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

DEMULTIPLEXER:

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

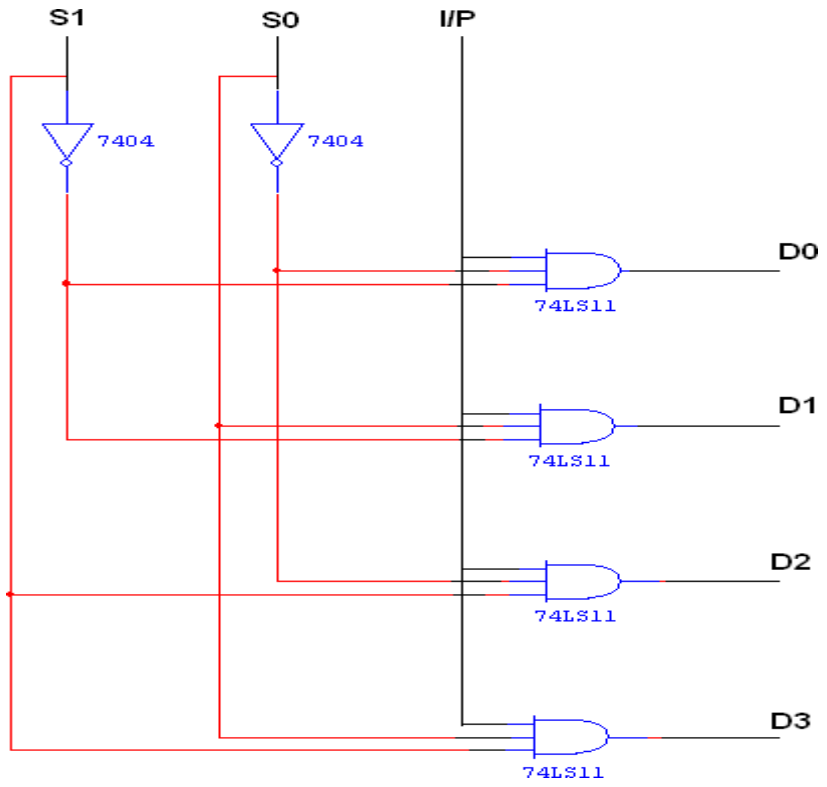
In the 1: 4 demultiplexer circuit, the data input line goes to all of the AND gates. The data select lines enable only one gate at a time and the data on the data input line will pass through the selected gate to the associated data output line.

1:4 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

LOGIC DIAGRAM FOR DEMULTIPLEXER:



FUNCTION TABLE:

SI	SO	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

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.

APPLICATION:

i. It can be used to implement logic functions in SOP form.

ii. It can be used to as a parallel to serial converter.

POST-LAB EXERCISE:

1. What is another name of demultiplexer?
2. What are the differences between a MUX & DEMUX?
3. How would you construct a logic function generator using multiplexers?
4. Draw logic symbol of a 4-to-1 multiplexer.
5. How many pins in IC74150?

RESULT:

Thus the multiplexer and De-multiplexer using logic gates were designed and implemented.

EXPT NO.:3 **DATE :**

DESIGN AND IMPLEMENTATION OF ENCODER AND DECODER

AIM:

To design and implement encoder and decoder using logic gates.

COMPONENTS 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 required

PRE-LAB EXERCISE:

1. What is meant by encoder?
2. What is meant by decoder?
3. What is priority encoder?
4. What are the applications of encoder and decoder?
5. What is BCD to seven segment decoder?

THEORY:

ENCODER:

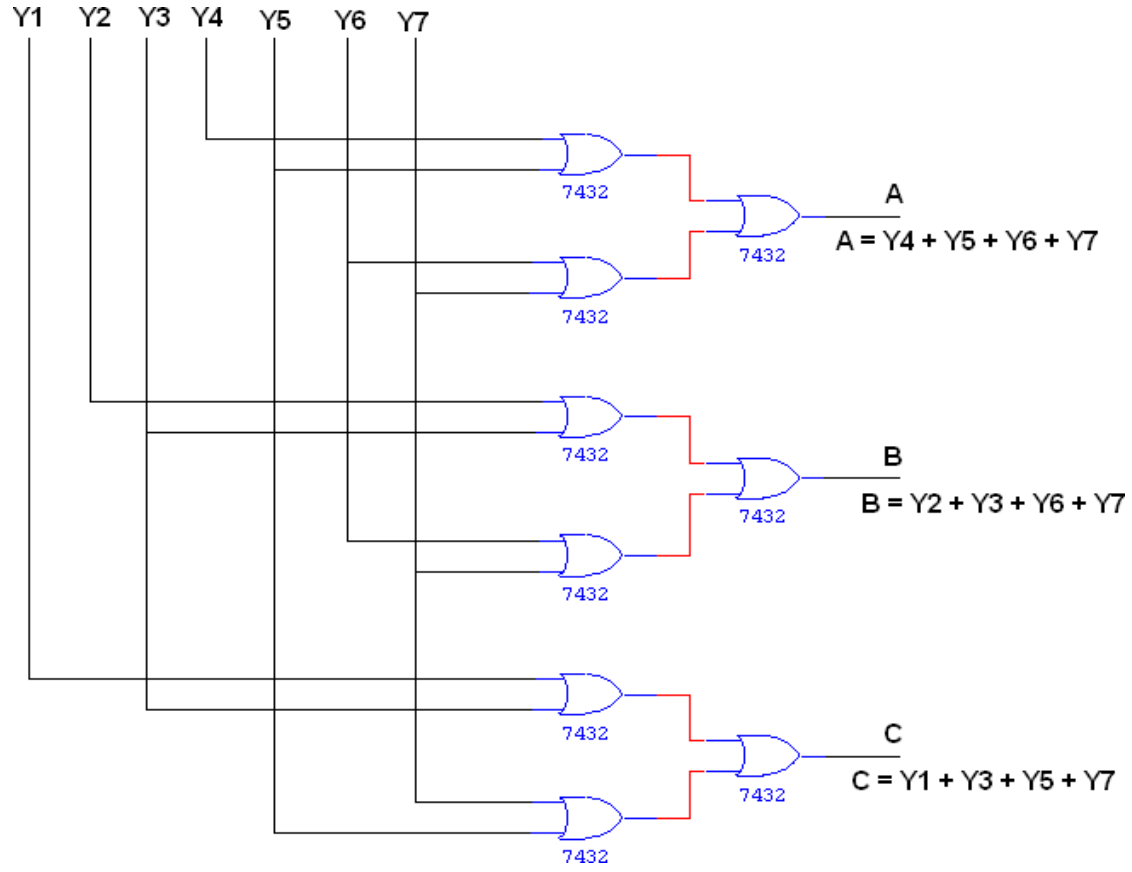
An encoder is a digital circuit that perform inverse operation of a decoder. An encoder has 2^n input lines and n output lines. In encoder the output lines generates the binary code corresponding to the input value. In octal to binary encoder it has eight inputs, one for each octal digit and three output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

ENCODER

TRUTH TABLE:

INPUT								OUTPUT		
Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	A	B	C
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

LOGIC DIAGRAM

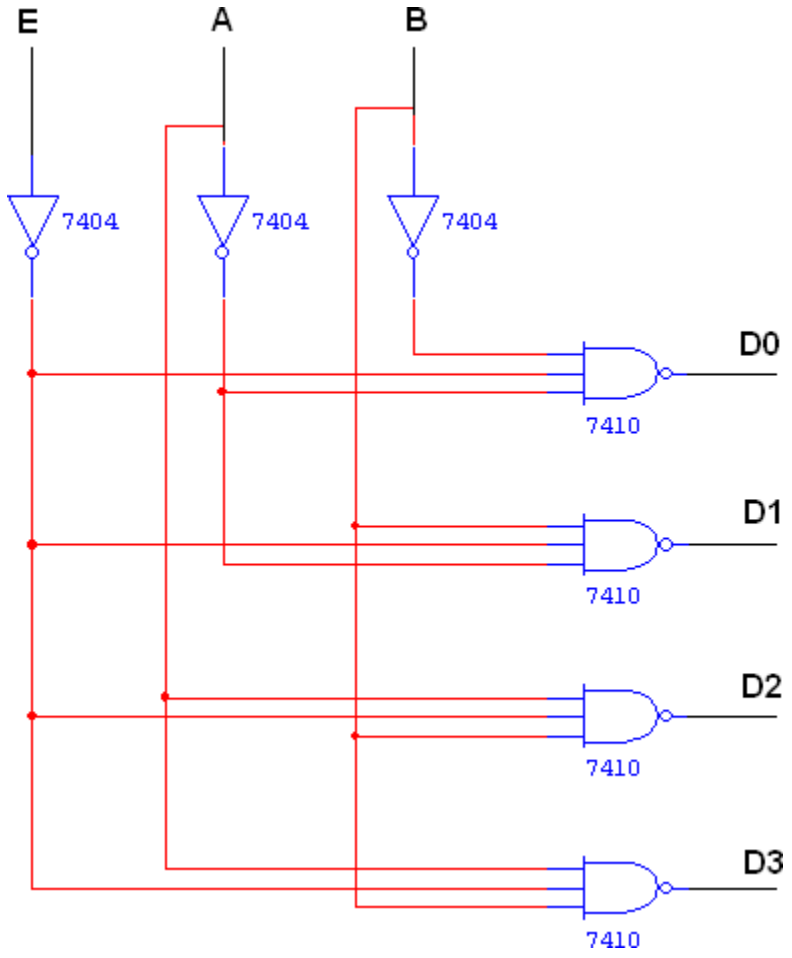


DECODER

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:



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 $2^n - 1$.

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.

POST-LAB EXERCISE:

1. Can more than one decoder output be activated at one time? Justify your answer.
2. What is the function of a decoder's enable input(s)?
3. How does an encoder differ from decoder?
4. How does a priority encoder differ from an ordinary encoder?
5. What is decimal to BCD encoder?

RESULT:

Thus the encoder and decoder using logic gates was designed and implemented.

EXPT NO.:4 **DATE :**

DESIGN AND IMPLEMENTATION OF 2 BIT MAGNITUDE COMPARATOR

Aim:

To design and implement a 2bit magnitude comparator using logic gates

COMPONENTS REQUIRED:

Sl. No.	COMPONENT	SPECIFICATION	QTY.
1.	EX-NOR	IC 747266	1
2.	OR GATE	IC 7432	1
3.	NOT GATE	IC 7404	1
4.	AND GATE	IC7408	1
5.	IC TRAINER KIT	-	1
6.	PATCH CORDS	-	As required

PRE-LAB EXERCISE:

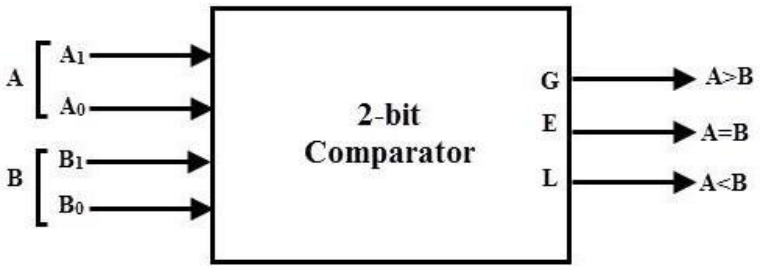
1. What is magnitude comparator?
2. Define most significant bit.
3. Why magnitude comparator is needed?
4. Which gate is a basic comparator?
5. How many inputs are required for a digital comparator?

THEORY:

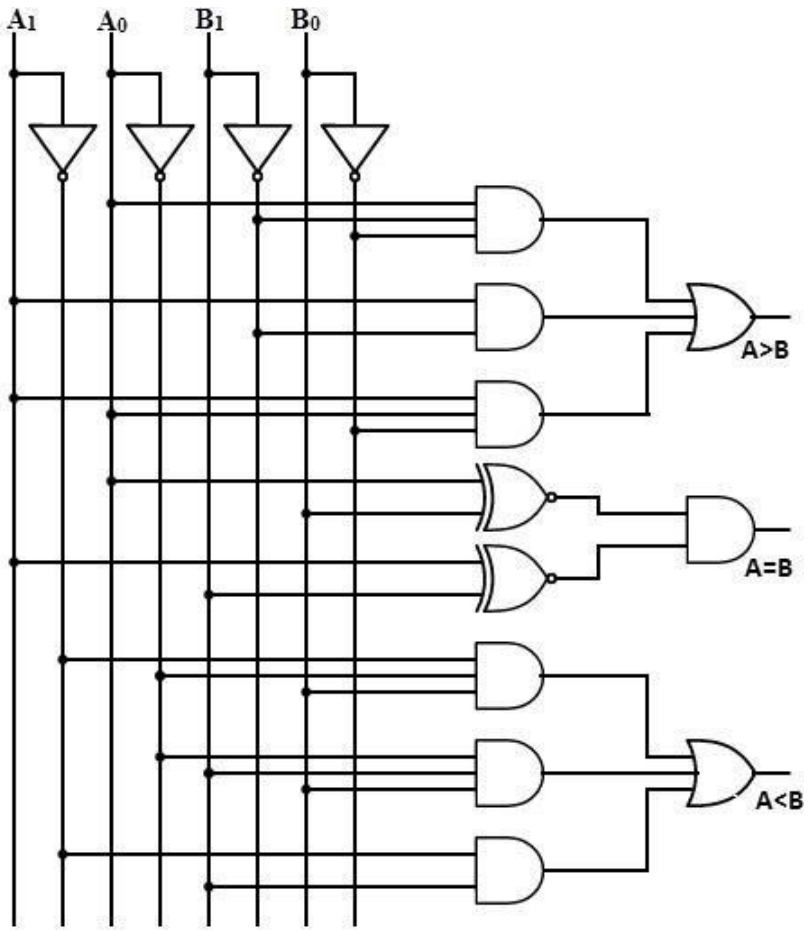
A 2-bit comparator compares two binary numbers, each of two bits and produces their relation such as one number is equal or greater than or less than the other. The figure below shows the block diagram of a two-bit comparator which has four inputs and three outputs.

The first number A is designated as A = A1A0 and the second number is designated as B = B1B0. This comparator produces three outputs as G (G = 1 if A>B), E (E = 1, if A = B) and L (L = 1 if A<B).

BLOCK DIAGRAM:



LOGIC DIAGRAM:



TRUTH TABLE:

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

K-MAP:

A>B

A_1A_0 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

A=B

A_1A_0 \ B_1B_0	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

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

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

$$= \overline{A1}\ \ \overline{B1}\,(\overline{A0}\ \ \overline{B0} + A0\,B0) + A1\,B1\,(A0\,B0 + \overline{A0}\ \ \overline{B0}\,)$$

$$= (A0\,B0 + \overline{A0}\ \ \overline{B0})\,(A1\,B1 + \ \overline{A1}\ \ \overline{B1})$$

$$= (A0\ \text{Ex-NOR}\ B0)\ (A1\ \text{Ex-NOR}\ B1)$$

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

By using above obtained Boolean equation for each output, the logic diagram can be implemented by using NOT gates, AND gates, OR gates and Ex-NOR gates.

POST- LAB EXERCISE:

1. If two numbers are not equal then binary variable will be….?
2. What is inequality?
3. Tell the applications of magnitude comparator.
4. How many types of digital comparators are there?
5. TTL 74LS85 is which type of magnitude comparator?

Result:

Thus the 2-bit magnitude comparator was designed and the values are compared.

EXPT NO.:5 **DATE:**

DESIGN AND IMPLEMENTATION OF COUNTERS USING FLIPFLOPS

AIM:

To design and verify 4 bit ripple counter.

COMPONENTS 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 required

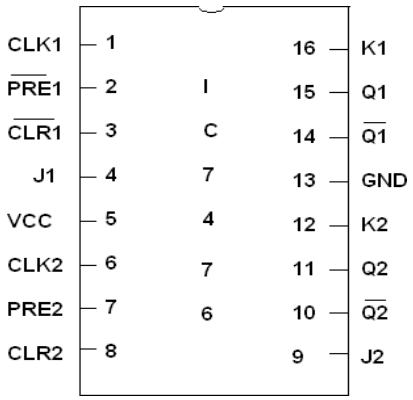
PRE-LAB EXERCISE:

1. What is counter?
2. What are the types of counter?
3. Distinguish between a ripple counter & a synchronous counter.
4. Define the modulus of a counter.
5. How is a modulus counter built using count reset?

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:

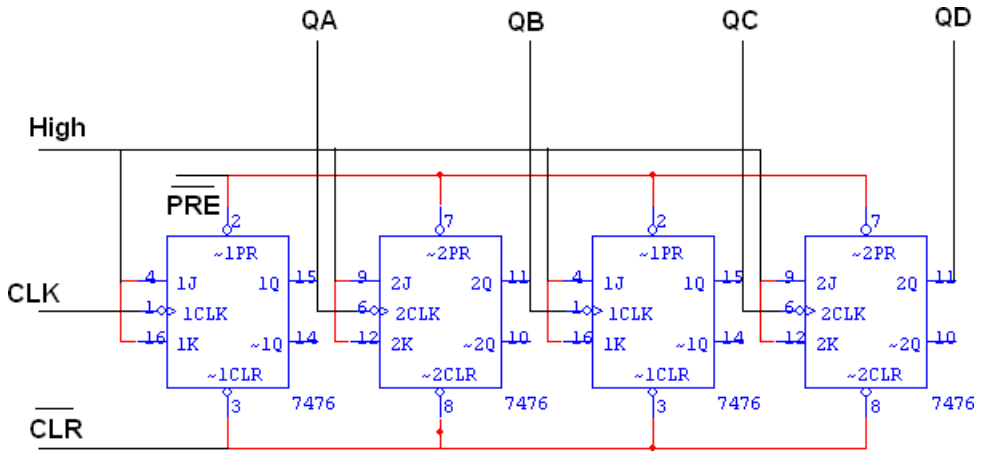


4 BIT RIPPLE COUNTER

TRUTH TABLE:

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

LOGIC DIAGRAM:



NOTE:

In 4 bit Ripple counter, Use One IC7476 for FF A & C and another One IC7476 for FF B & D.

POST-LAB EXERCISE:

1. What is an in asynchronous sequential circuit?
2. What is a sequence generator?
3. What is an asynchronous decade counter?
4. Define synchronous counter.
5. What is Johnson counter

RESULT:

Thus the 4 bit ripple counter and Mod-10 /Mod-12 ripple counter were constructed and their state tables were verified.

EXPT NO.:6 **DATE:**

STUDY OF SHIFT REGISTERS IN VARIOUS MODE OF OPERATION

AIM:

To study and implement shift register in various modes of operation

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

APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	D flip flop	IC 7474	2
2.	OR gate	IC 7432	1
3.	IC Trainer kit	-	1
4.	Patch cords	-	As required

THEORY:

SHIFT REGISTER:

The Shift Register is another type of sequential logic circuit that is used for the storage or transfer of data in the form of binary numbers and then "shifts" the data out once every clock cycle, hence the name "shift register". It basically consists of several single bit "D-Type Data Latches", one for each bit (0 or 1) connected together in a serial or daisy-chain arrangement so that the output from one data latch becomes the input of the next latch and so on.

The data bits may be fed in or out of the register serially, i.e. one after the other from either the left or the right direction, or in parallel, i.e. all together. The number of individual data latches required to make up a single Shift Register is determined by the number of bits to be stored with the most common being 8-bits (one byte) wide, i.e. eight individual data latches. The individual data latches that make up a single shift register are all driven by a common clock (Clk) signal making them synchronous devices.

Shift register IC's are generally provided with a *"*clear" or "reset" connection so that they can be "SET" or "RESET" as required. Generally, shift registers operate in one of four different modes with the basic movement of data through a shift register being:

Serial-in to Parallel-out (SIPO):

The register is loaded with serial data, one bit at a time, with the stored data being available in parallel form.

Serial-in to Serial-out (SISO):

The data is shifted serially "IN" and "OUT" of the register, one bit at a time in either a left or right direction under clock control.

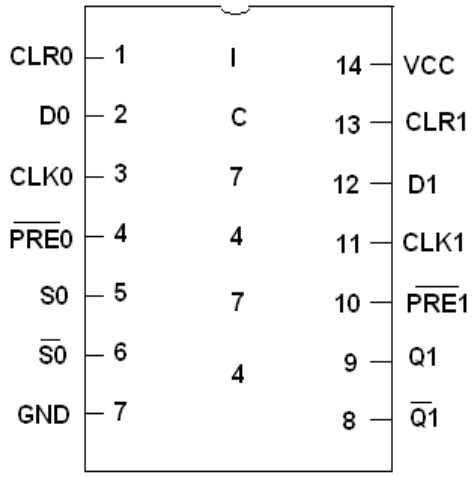
Parallel-in to Serial-out (PISO):

The parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.

Parallel-in to Parallel-out (PIPO):

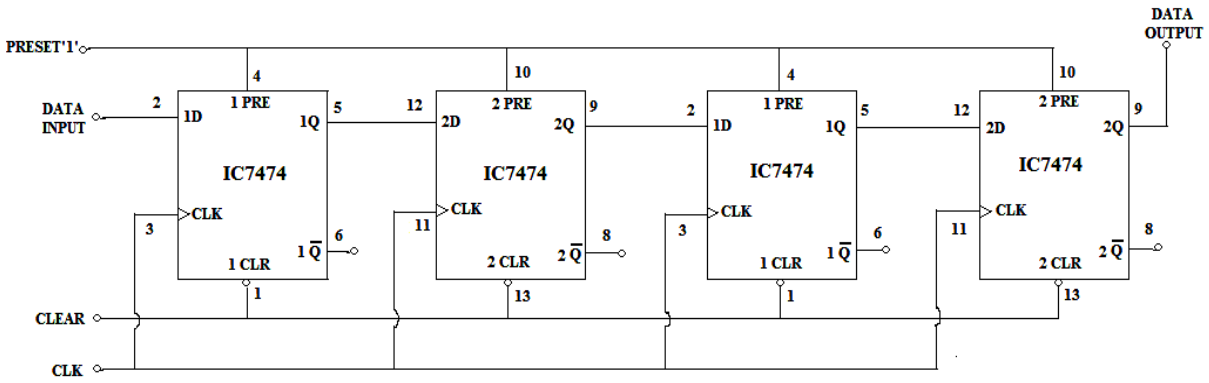
The parallel data is loaded simultaneously into the register, and transferred together to their respective outputs by the same clock pulse.

PIN DIAGRAM:



LOGIC DIAGRAM:

SERIAL IN SERIAL OUT:

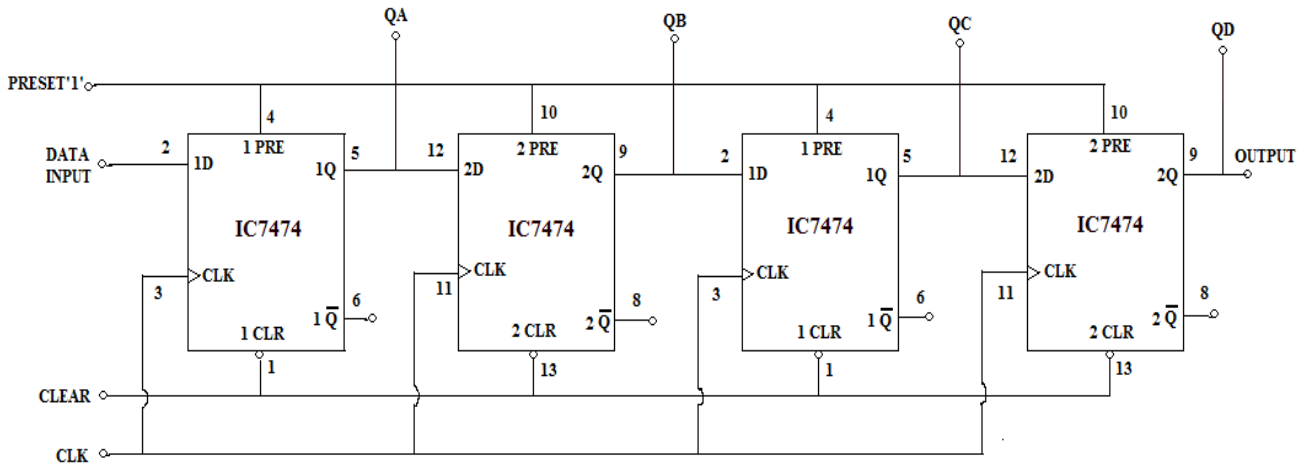


TRUTH TABLE:

CLK	Serial inputs	Serial outputs
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:

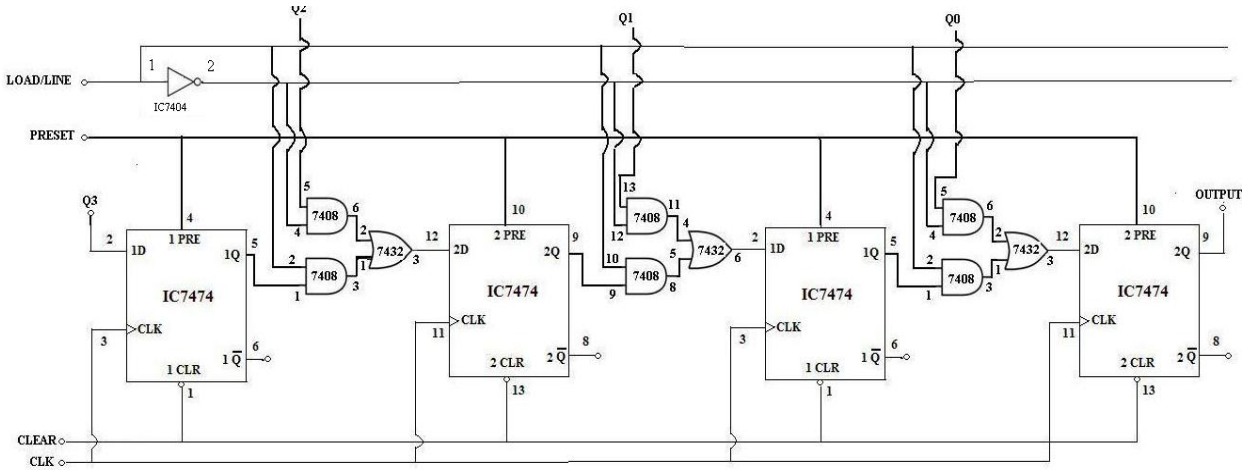


TRUTH TABLE:

CLK	Serial inputs	Parallel Outputs			
		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	0
4	1	1	0	0	1

LOGIC DIAGRAM:

PARALLEL IN SERIAL OUT:

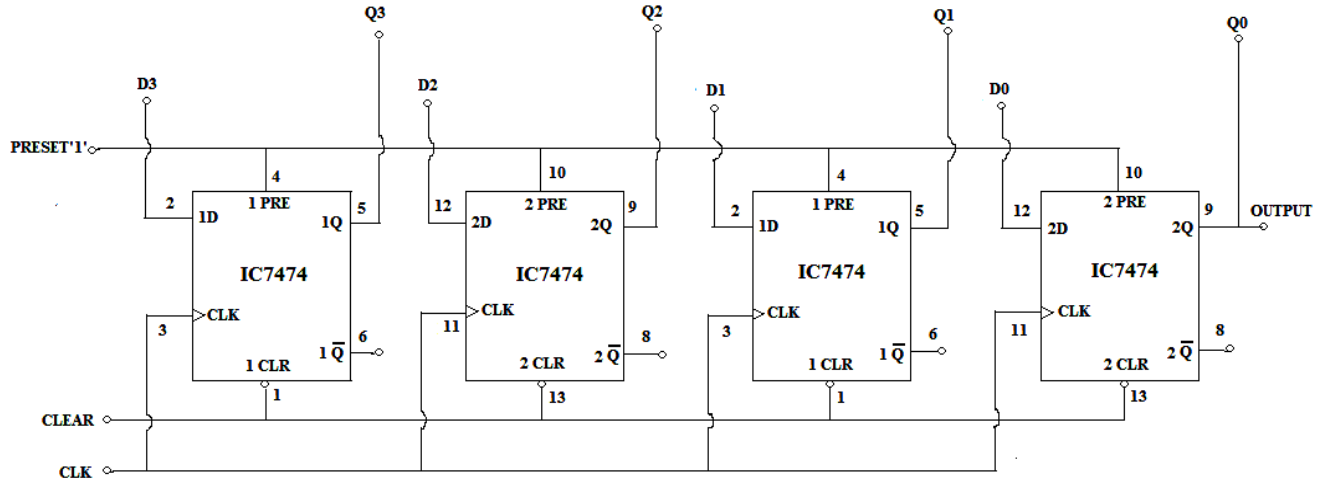


TRUTH TABLE:

CLK	Parallel Inputs				Serial Output
	Q3	Q2	Q1	Q0	
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	Parallel Inputs				Parallel Outputs			
	D_0	D_1	D_2	D_3	Q_0	Q_1	Q_2	Q_3
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:

Thus, shift registers are implemented and its operations are verified using IC7474.

EXPT NO.:7 **DATE :**

HALF & FULL ADDER USING VHDL

AIM:

To write VHDL code and study the functionality of Half Adder and Full Adder.

SOFTWARE REQUIRED:

Xilinx ISE Design Studio

HALF ADDER:

Verilog Code:

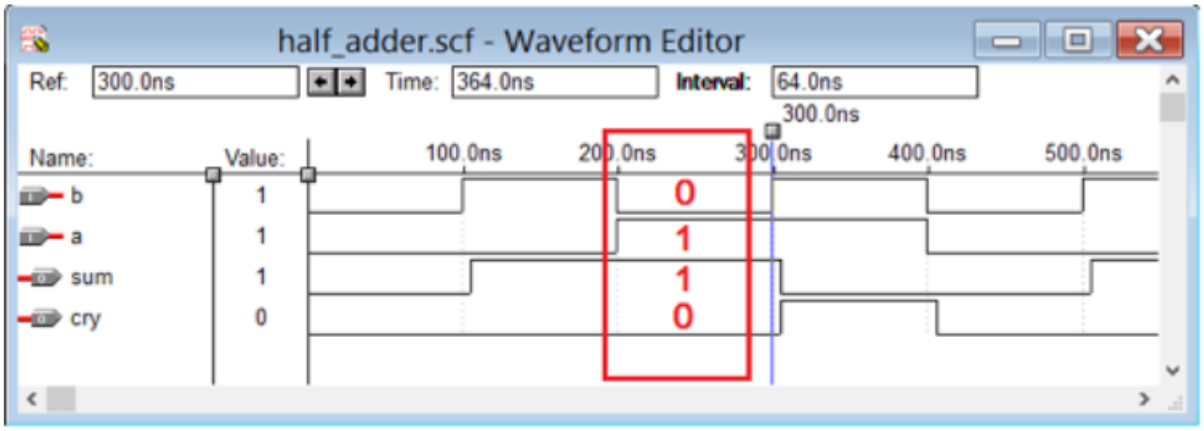
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ha is
    Port ( i1_ha : in  STD_LOGIC;
           i2_ha : in  STD_LOGIC;
           sum_ha : out  STD_LOGIC;
           carry_ha : out  STD_LOGIC);
end ha;

architecture ha_arch of ha is

begin
sum_ha <= i1_ha xor i2_ha;
carry_ha <= i1_ha and i2_ha;
end ha_arch;
```

Simulation Waveform:



Full Adder:

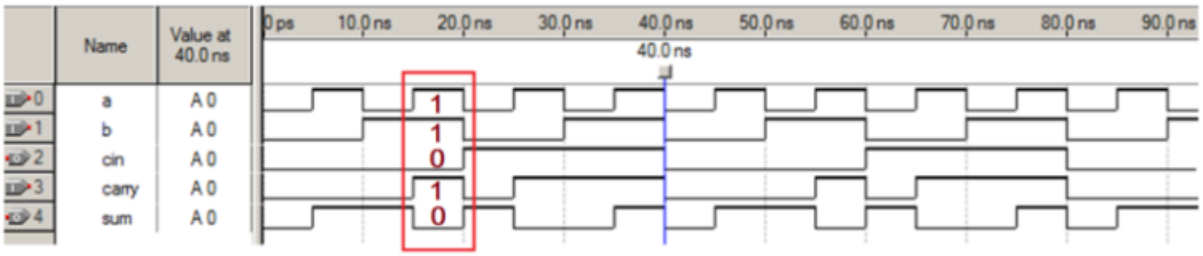
Verilog Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity fa is
    Port ( i1_fa, i2_fa, i3_fa : in  STD_LOGIC;
           sum_fa, carry_fa : out  STD_LOGIC);
end fa;

architecture fa_arch of fa is
component ha
   port(i1_ha, i2_ha : in std_logic;
        sum_ha, carry_ha : out std_logic);
end component;
component or_2
   port(i1_or2, i2_or2 : in std_logic;
        o_or2: out std_logic);
end component;
signal temp1,temp2,temp3 : std_logic;
begin
u1:ha port map(i1_ha => i1_fa, i2_ha => i2_fa, sum_ha => temp1, carry_ha => temp2);
u2:ha port map(i1_ha => temp1, i2_ha => i3_fa, sum_ha => sum_fa, carry_ha => temp3);
u3:or_2 port map(i1_or2 => temp3, i2_or2 => temp2, o_or2 => carry_fa);
end fa_arch;
```

Simulation Waveform:



RESULT:

Thus, half adder and full adder is studied using the VHDL code.

EXPT NO.:8 **DATE :**

D-FLIP FLOP USING VHDL

AIM:

To write VHDL code and study the functionality of D- Flip Flop

SOFTWARE REQUIRED:

Xilinx ISE Design Studio

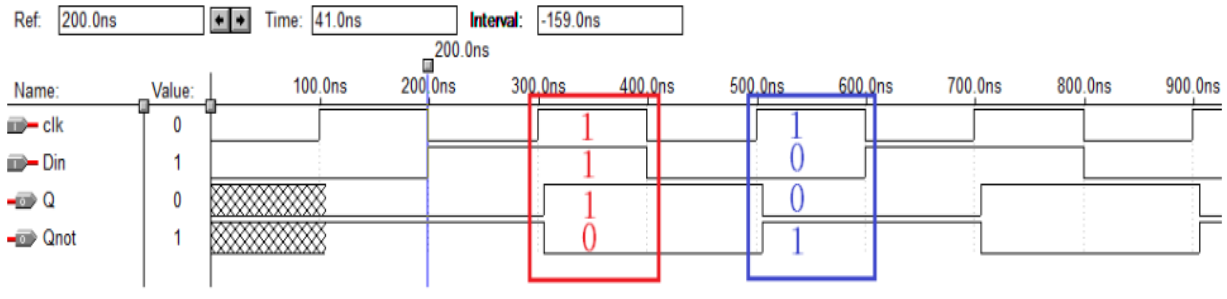
Verilog Code:

```
library ieee;
use ieee.std_logic_1164.all;

entity D_FF is
        port  ( D : in std_logic;
                CLK_D          : in std_logic;
                Q : out std_logic);
end D_FF;

architecture D_FF_arch of D_FF is
begin
        process (CLK_D)
        begin
                if (CLK_D'event and CLK_D = '1') then
                Q <= D;
                end if;
        end process;
end D_FF_arch;
```

Simulation Waveform:



RESULT:

Thus, D- flip flop is studied using the VHDL code