MAHENDRA COLLEGE OF ENGINEERING

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

CS 6211 & IT 6211 - DIGITAL LAB

LAB MANUAL

Branch : **CSE & IT**
Year/semester : **I Year / II Semester**

Syllabus

CS6211 DIGITAL LABORATORY

OBJECTIVES: The student should be made to:

- Understand the various logic gates.
- Be familiar with various combinational circuits.
- Understand the various components used in the design of digital computers.
- Be exposed to sequential circuits
- Learn to use HDL

LIST OF EXPERIMENTS:

1. Verification of Boolean Theorems using basic gates.
2. Design and implementation of combinational circuits using basic gates for arbitrary functions, code converters.
3. Design and implementation of combinational circuits using MSI devices:
   - 4 – bit binary adder / subtractor
   - Parity generator / checker
   - Magnitude Comparator
   - Application using multiplexers
4. Design and implementation of sequential circuits:
   - Shift –registers
   - Synchronous and asynchronous counters
5. Coding combinational / sequential circuits using HDL.
6. Design and implementation of a simple digital system (Mini Project).

TOTAL: 45 PERIODS

OUTCOMES: At the end of this course, the student will be able to:

- Use boolean simplification techniques to design a combinational hardware circuit.
- Design and Implement combinational and sequential circuits.
- Analyze a given digital circuit – combinational and sequential.
- Design the different functional units in a digital computer system.
- Design and Implement a simple digital system.

LABORATORY REQUIREMENT FOR BATCH OF 30 STUDENTS

HARDWARE:

1. Digital trainer kits 30

2. Digital ICs required for the experiments in sufficient numbers 96

SOFTWARE:

1. HDL simulator.

DO'S

1. **Be regular to the lab.**
2. **Follow proper Dress Code.**
3. **Maintain Silence.**
4. **Know the theory behind the experiment before coming to the lab.**
5. **Identify the different leads or terminals or pins of the IC before making connection.**
6. **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.**
7. **Know the Current and Voltage rating of the IC's before using them in the experiment.**
8. **Avoid unnecessary talking while doing the experiment.**
9. **Handle the IC Trainer Kit properly.**
10. **Mount the IC Properly on the IC Zif Socket.**
11. **Handle the microprocessor kit properly.**
12. **While doing the Interfacing, connect proper voltages to the interfacing kit.**
13. **Keep the Table clean.**
14. **Take a signature of the In charge before taking the kit/components.**
15. **After the completion of the experiments switch off the power supply and return the apparatus.**

DON' TS

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 µ p/IC trainer kits unnecessarily.
7. Do not panic if you don't get the output.

The Laboratory Notebook:

Each student must have their own laboratory notebook. All pre-lab exercises and laboratory reports are to be entered into your notebook.

Your notebook must be clearly labeled on the cover with the following information:

Module **:** Digital Lab
Name **:**
Register No **:**
Lab Partner Name **:**

Introduction

There are 3 hours allocated to a laboratory session in Digital Electronics. It is a necessary part of the course at which attendance is compulsory.

Here are some guidelines to help you perform the experiments and to submit the reports:

1. Read all instructions carefully and carry them all out.
2. Ask a demonstrator if you are unsure of anything.
3. Record actual results (comment on them if they are unexpected!)
4. Write up full and suitable conclusions for each experiment.
5. If you have any doubt about the safety of any procedure, contact the demonstrator beforehand.
6. **THINK** about what you are doing!

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.

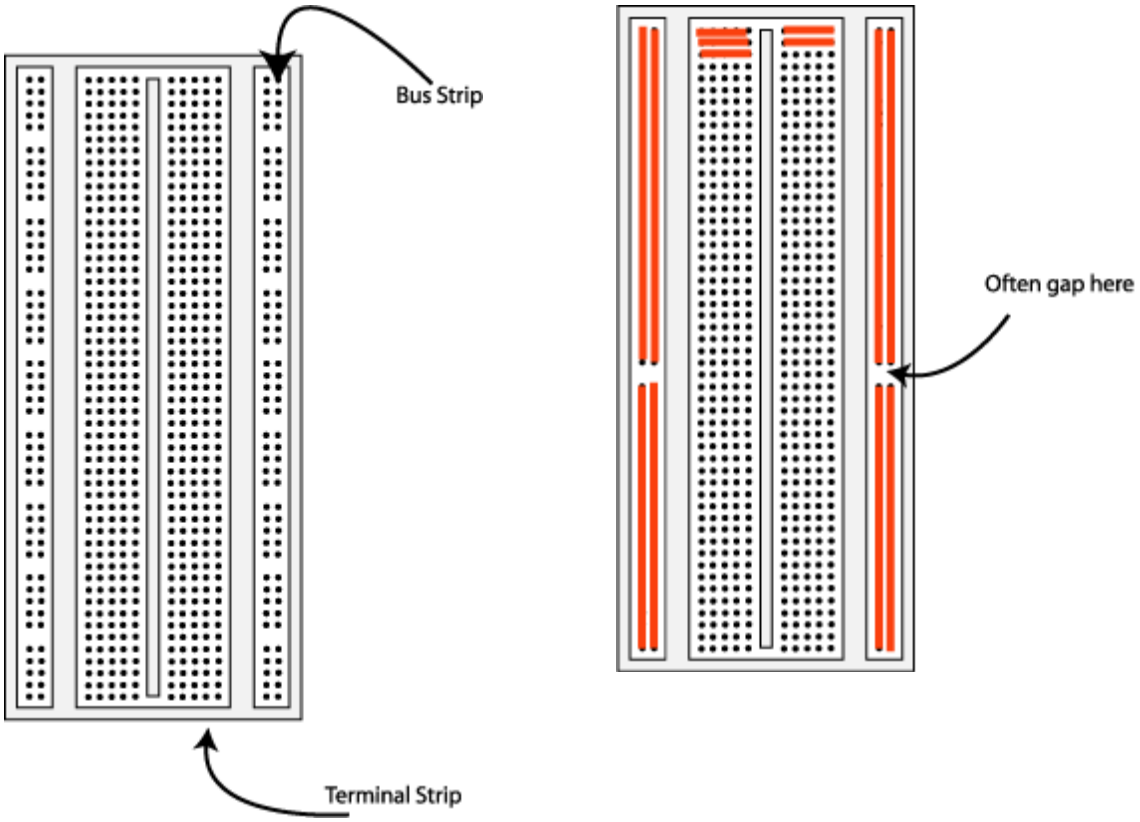


Fig 1. The breadboard. The lines indicate connected holes.

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

Building the Circuit:

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

1. Turn the power (Trainer Kit) off before you build anything!

2. Make sure the power is off before you build anything!

3. Connect the +5V and ground (GND) leads of the power supply to the power and ground bus strips on your breadboard.

4. Plug the chips you will be using into the breadboard. Point all the chips in the same direction with pin 1 at the upper-left corner. (Pin 1 is often identified by a dot or a notch next to it on the chip package)

5. Connect +5V and GND pins of each chip to the power and ground bus strips on the breadboard.

6. Select a connection on your schematic and place a piece of hook-up wire between corresponding pins of the chips on your breadboard. It is better to make the short connections before the longer ones. Mark each connection on your schematic as you go, so as not to try to make the same connection again at a later stage.

7. Get one of your group members to check the connections, **before you turn the power on**.

8. If an error is made and is not spotted before you turn the power on. Turn the power off immediately before you begin to rewire the circuit.

9. At the end of the laboratory session, collect you hook-up wires, chips and all equipment and return them to the demonstrator.

10. Tidy the area that you were working in and leave it in the same condition as it was before you started.

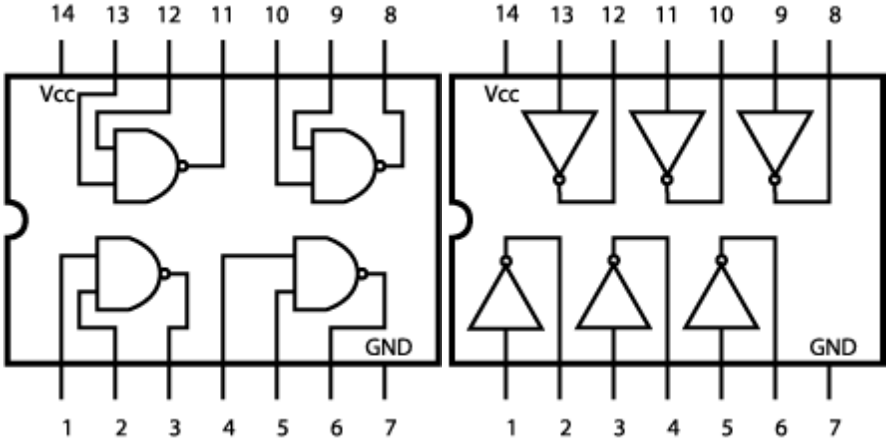
Common Causes of Problems

1. Not connecting the ground and/or power pins for all chips.
2. Not turning on the power supply before checking the operation of the circuit.
3. Leaving out wires.
4. Plugging wires into the wrong holes.
5. Driving a single gate input with the outputs of two or more gates
6. Modifying the circuit with the power on.

In all experiments, you will be expected to obtain all instruments, leads, components at the start of the experiment and return them to their proper place after you have finished the experiment. Please inform the demonstrator or technician if you locate faulty equipment. If you damage a chip, inform a demonstrator, don't put it back in the box of chips for somebody else to use.

Example Implementation of a Logic Circuit

Build a circuit to implement the Boolean function F = /(/A./B), please note that the notation **/A** refers to $\overline{A}$. You should use that notation during the write-up of your laboratory experiments.



Quad 2 Input 7400 Hex 7404 Inverter

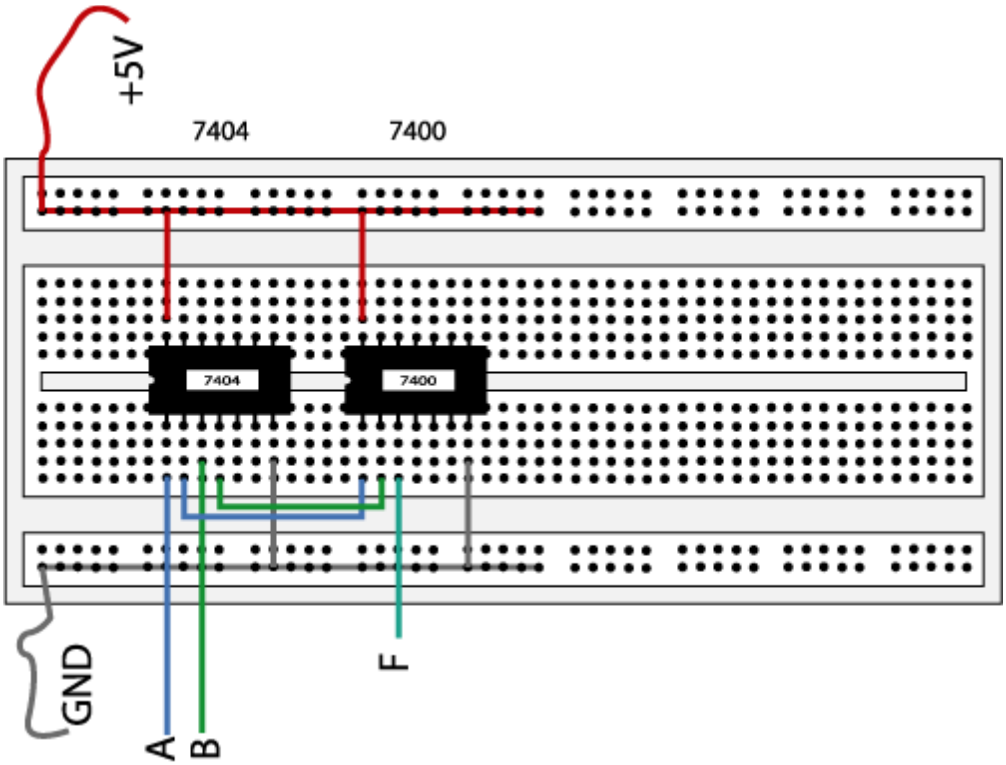
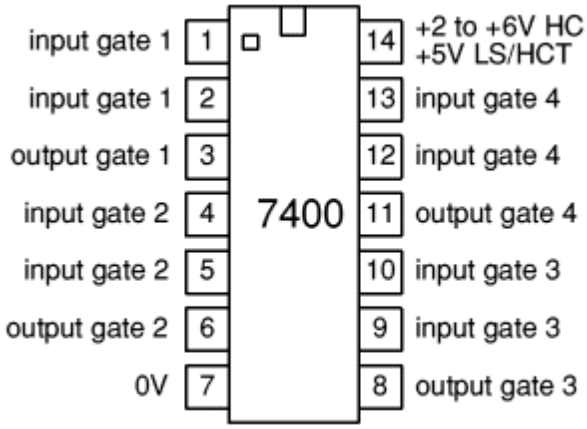


Fig 2. The complete designed and connected circuit

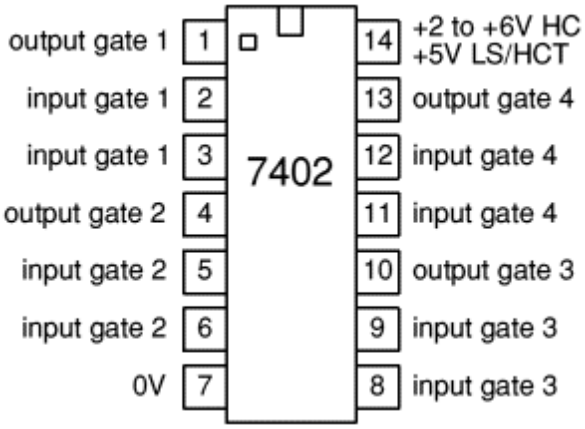
Sometimes the chip manufacturer may denote the first pin by a small indented circle above the first pin of the chip. Place your chips in the same direction, to save confusion at a later stage. Remember that you must connect power to the chips to get them to work.

Useful IC Pin details

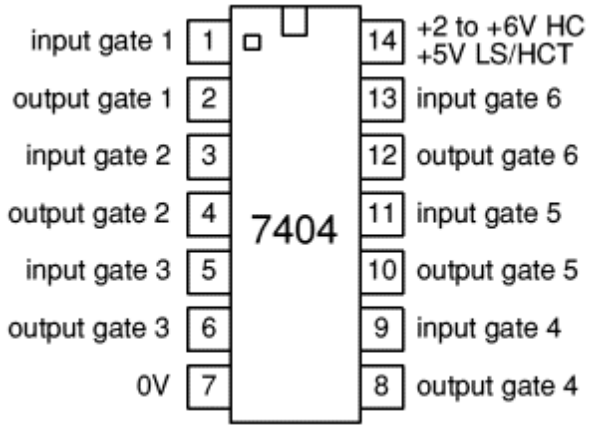
7400(NAND)



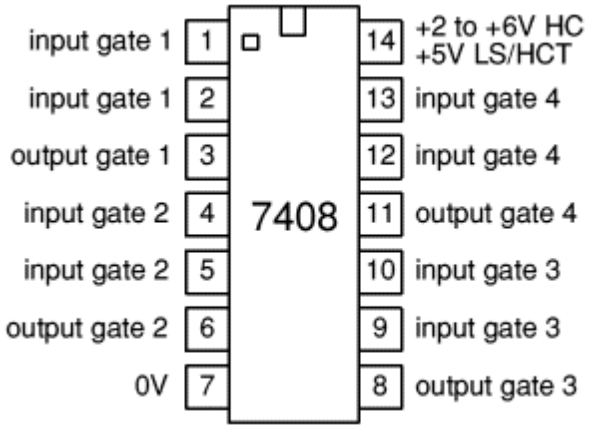
7402(NOR)



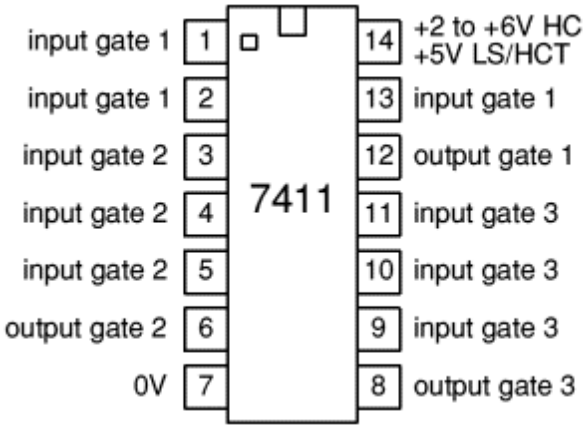
7404(NOT)



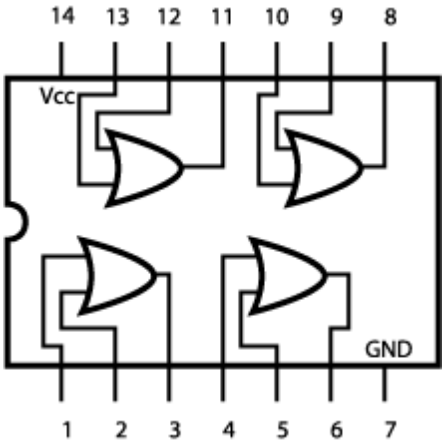
7408(AND)



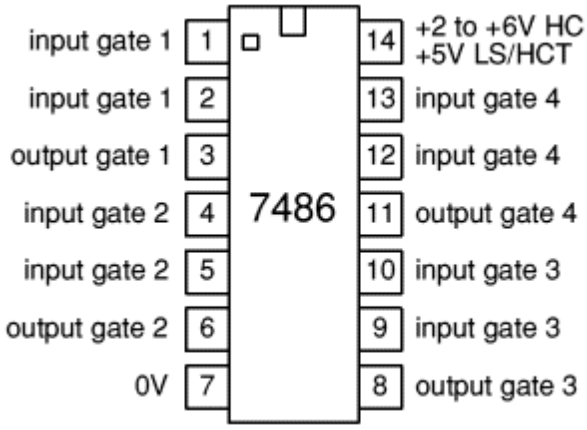
7411(3-i/p AND)

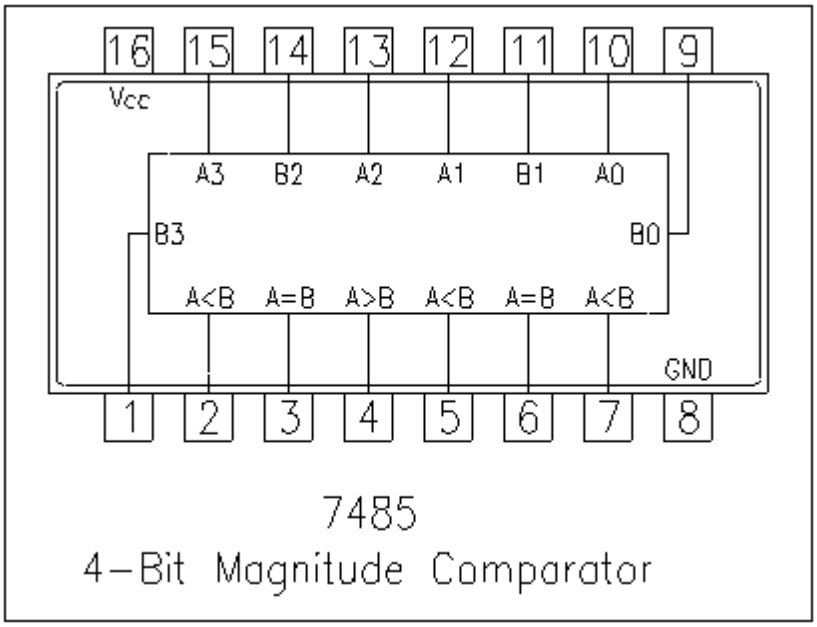


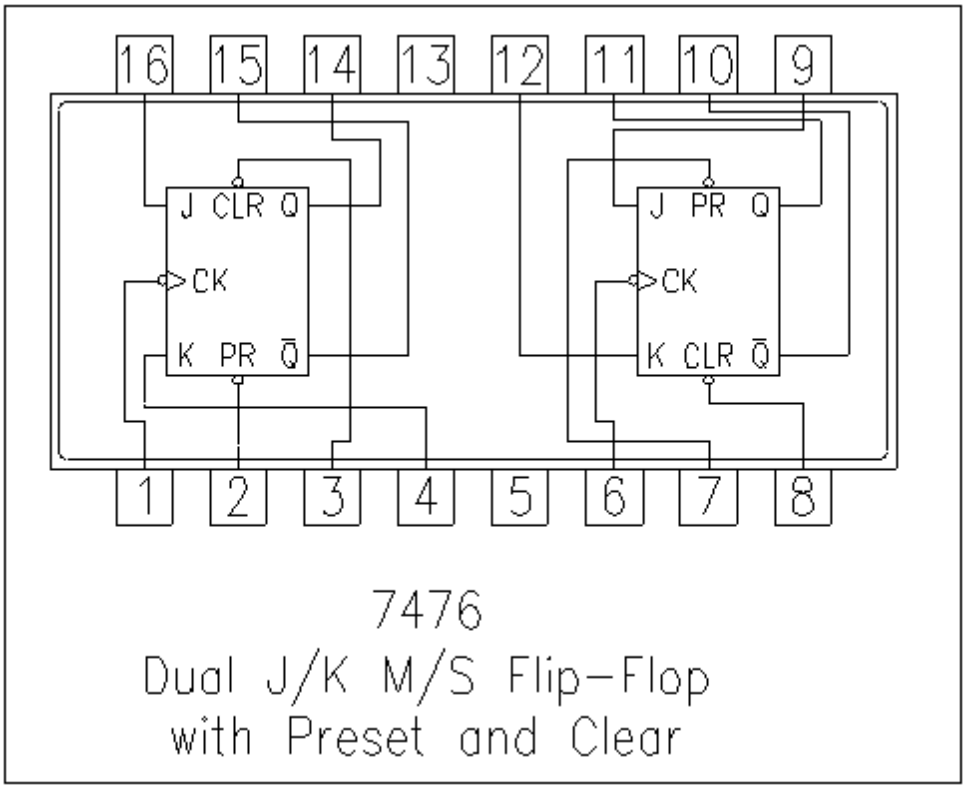
7432(OR)



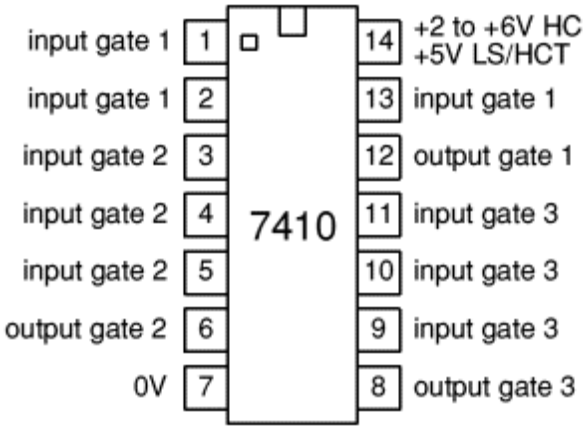
7486(EX-OR)


input gate 1
1
input gate 1
2
output gate 1
3
input gate 2
4
input gate 2
5
output gate 2
6
0V
7
7486
14
+2 to +6V HC
+5V LS/HCT
13
input gate 4
12
input gate 4
11
output gate 4
10
input gate 3
9
input gate 3
8
output gate 3

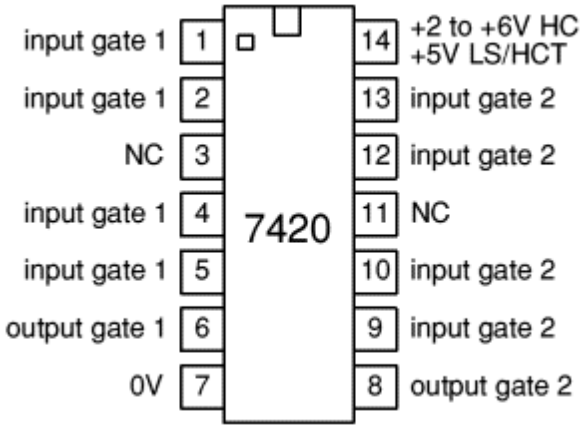

16
15
14
13
12
11
10
9
Vcc
A3
B2
A2
A1
B1
A0
B3
B0
A<B
A=B
A>B
A<B
A=B
A<B
GND
1
2
3
4
5
6
7
8
7485
4-Bit Magnitude Comparator



7410(3-i/p NAND)



7420(4-i/p NAND)

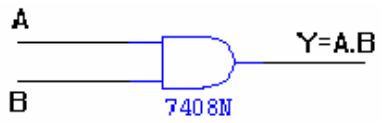


CS6211 - DIGITAL LAB

LIST OF EXPERIMENTS

AND GATE:

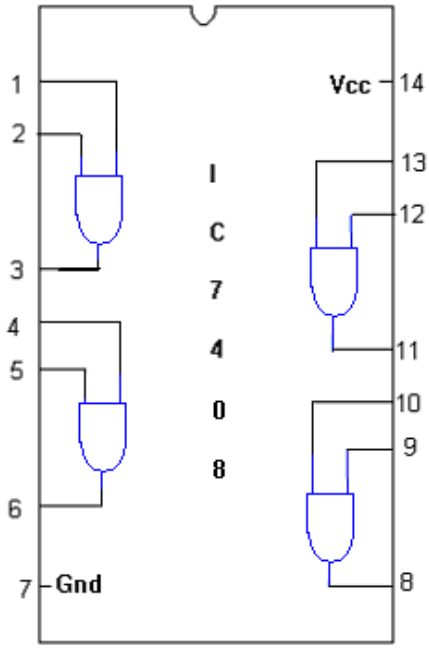
SYMBOL:



TRUTH TABLE

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

PIN DIAGRAM:



EXP NO: 1 STUDY OF LOGIC GATES

AIM:

To study about logic gates and verify their truth tables.

APPARATUS REQUIRED:

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

THEORY:

Circuit that takes the logical decision and the process are called logic gates. Each gate has one or more input and only one output. OR, AND and NOT are basic gates. NAND, NOR and X-OR are known as universal gates. Basic gates form these gates.

AND GATE:

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

OR GATE:

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

NOT GATE:

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

NAND GATE:

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

NOR GATE:

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

X-OR GATE:

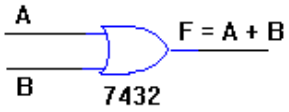
The output is high when any one of the inputs is high. The output is low when both the inputs are low and both the inputs are high.

PROCEDURE:

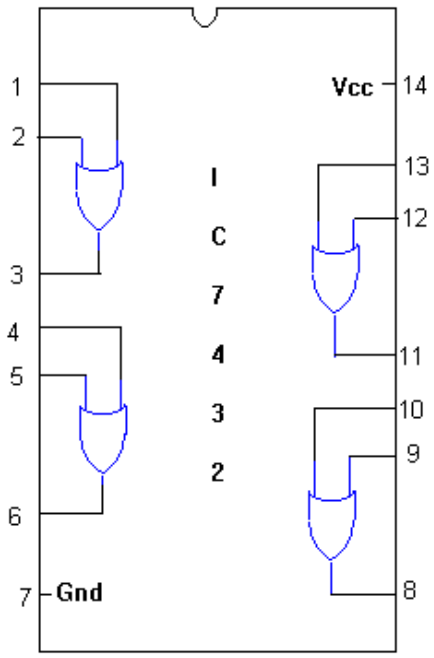
1. Connections are given as per circuit diagram.
2. Logical inputs are given as per circuit diagram.
3. Observe the output and verify the truth table.

OR GATE:

SYMBOL :



PIN DIAGRAM :

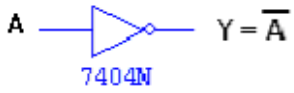


TRUTH TABLE

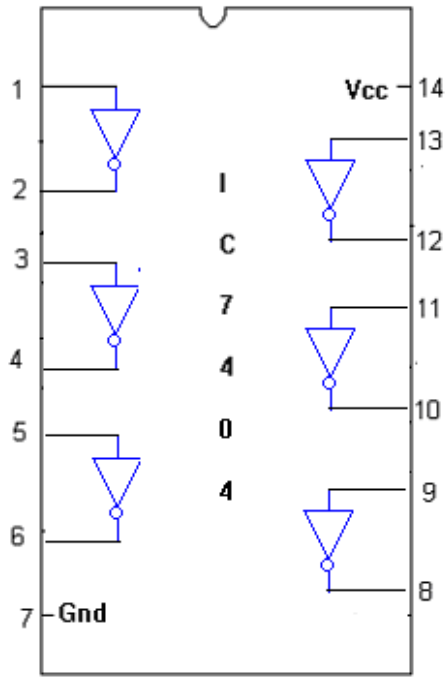
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

NOT GATE:

SYMBOL:



PIN DIAGRAM:

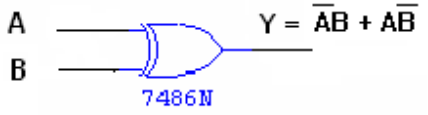


TRUTH TABLE :

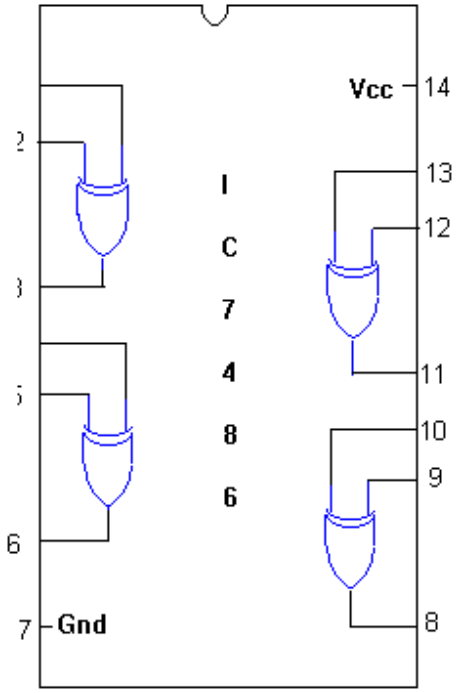
A	$\overline{A}$
0	1
1	0

X-OR GATE :

SYMBOL :



PIN DIAGRAM :

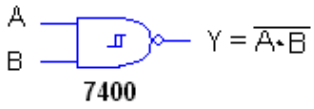


TRUTH TABLE :

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

2-INPUT NAND GATE:

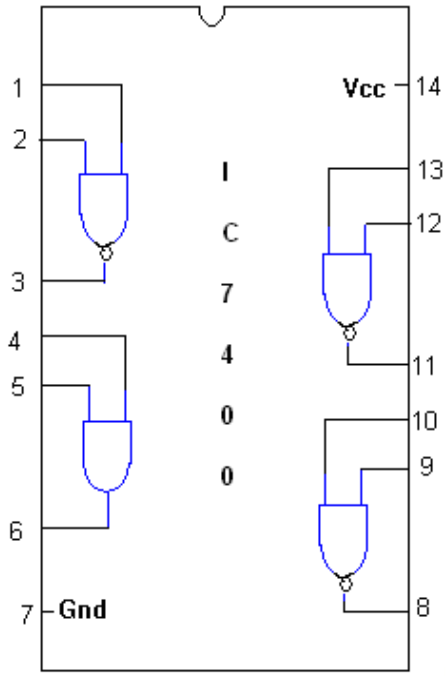
SYMBOL:



TRUTH TABLE

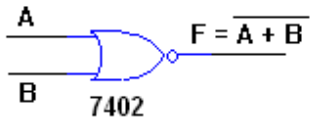
A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

PIN DIAGRAM:



NOR GATE:

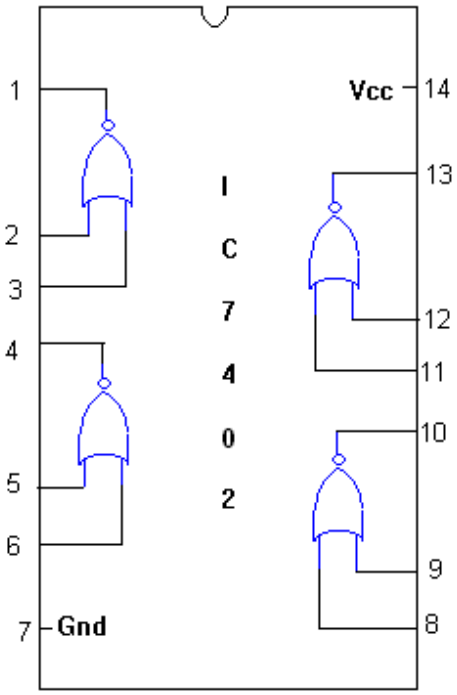
SYMBOL :



TRUTH TABLE

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

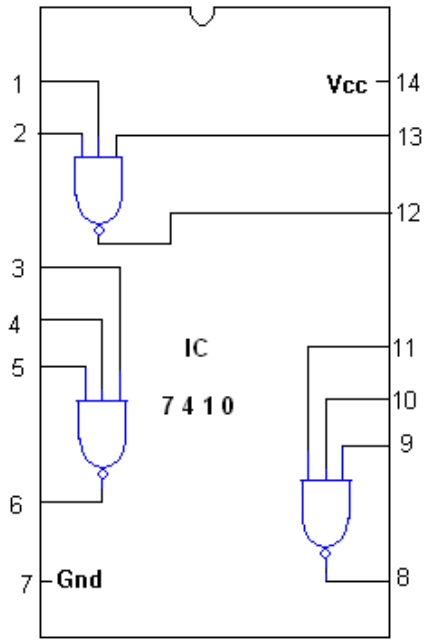
PIN DIAGRAM :



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

RESULT:

Thus the logic gates with the help of truth table are studied and verified.

VIVA QUESTIONS

EXP NO: 2 VERIFICATION OF BOOLEAN THEOREMS USING LOGIC GATES

AIM:

To design the logic circuits and verify its truth table for the following Boolean functions.

APPARATUS REQUIRED:

Sl. No.	COMPONENTS	SPECIFICATION/ RANGE	Qty
1.	AND GATE	IC 7408	1
2.	OR GATE	IC 7432	1
3.	NOT GATE	IC 7404	1
4.	DIGITAL TRAINER KIT	-	1
5.	CONNECTING WIRES	-	Req.

THEORY:

Boolean theorems can help us to simplify logic expressions and logic circuits. Boolean algebra is an individual tool in describing, analyzing, designing and implementing digital circuits.

BASIC BOOLEAN LAWS

1. Commutative Law

The binary operator OR, AND is said to be commutative if,

1. A+B = B+A
2. A.B=B.A

2. Associative Law

The binary operator OR, AND is said to be associative if,

1. A+(B+C) = (A+B)+C
2. A.(B.C) = (A.B).C

3. Distributive Law

The binary operator OR, AND is said to be distributive if,

1. A+(B.C) = (A+B).(A+C)
2. A.(B+C) = (A.B)+(A.C)

4. Absorption Law

1. A+AB = A

2. $A+\overline{A}B = A+B$

5. Involution (or) Double complement Law

1. $\overline{\overline{A}} = A$

6. Idempotent Law

1. A+A = A
2. A.A = A

7. Complementary Law

1. A+A' = 1
2. A.A' = 0

8. De Morgan's Theorem

1. The complement of the sum is equal to the sum of the product of the individual complements.

$\overline{A+B} = \overline{A}.\overline{B}$

2. The complement of the product is equal to the sum of the individual complements.

$\overline{A.B} = \overline{A}+\overline{B}$

9. Consensus Theorem

Consensus theorem is used to simplify the Boolean expression by eliminating the redundant terms.

AB+A'C+BC = AB+A'C

Proof: AB+A'C+BC = AB+A'C+BC (A+A')
= **AB**+A'C+**ABC**+A'BC
= AB(1+C)+A'C(1+B)
= AB+A'C. (using OR law, 1+C=1 and 1+B=1)

Design:

Associative Law:

Law 1:

A+ (B+C) = (A+B) +C

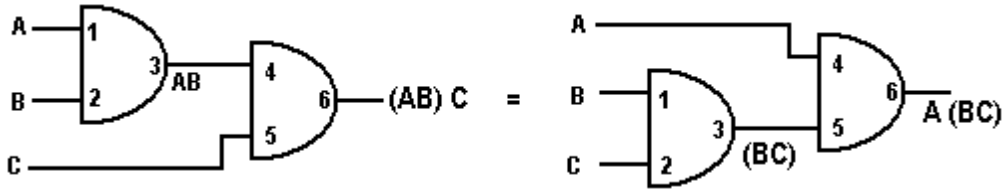
LOGIC DIAGRAM:

TRUTH TABLE:

INPUTS			OUTPUTS			
A	**B**	**C**	**A+B**	**(A+B)+C**	**B+C**	**A+(B+C)**
0	**0**	**0**				
0	**0**	**1**				
0	**1**	**0**				
0	**1**	**1**				
1	**0**	**0**				
1	**0**	**1**				
1	**1**	**0**				
1	**1**	**1**				

LAW 2: **(AB) C = A (BC)**

LOGIC DIAGRAM:



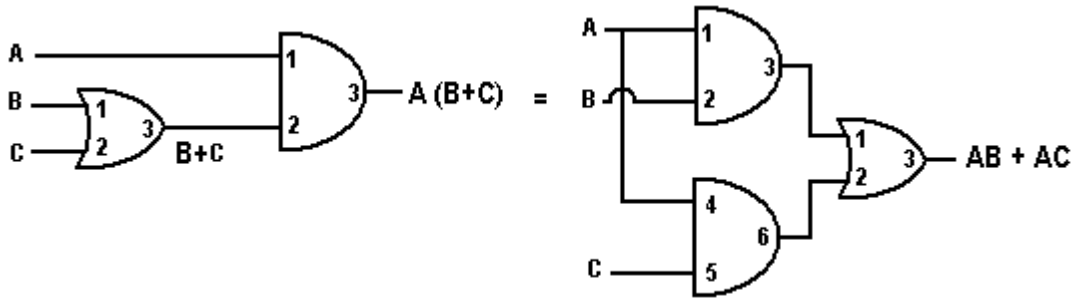
TRUTH TABLE:

INPUTS			OUTPUTS			
A	**B**	**C**	**A.B**	**(A.B).C**	**B.C**	**A.(B.C)**
0	**0**	**0**				
0	**0**	**1**				
0	**1**	**0**				
0	**1**	**1**				
1	**0**	**0**				
1	**0**	**1**				
1	**1**	**0**				
1	**1**	**1**				

Distributive Law:

A (B+C) = AB + AC

LOGIC DIAGRAM:

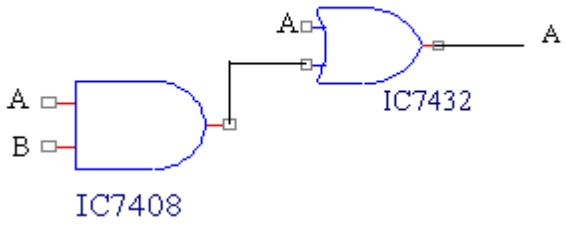


TRUTH TABLE:

INPUTS			OUTPUTS				
A	B	C	B+C	A(B+C)	AB	AC	AB+AC
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

Absorption Law:

A+AB = A

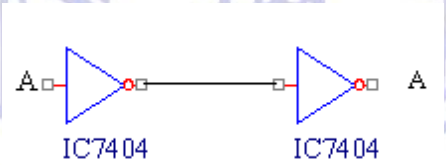


TRUTH TABLE:

INPUTS		OUTPUTS	
A	**B**	**A.B**	**A+(AB) i.e A**
0	**0**		
0	**0**		
0	**1**		
0	**1**		
1	**0**		
1	**0**		

Involution (or) Double complement Law:

$A = \overline{\overline{A}}$

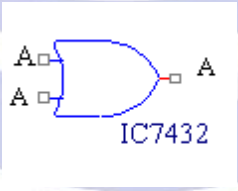


TRUTH TABLE:

INPUTS	OUTPUTS	
A	$\overline{A}$	$\overline{\overline{A}}$ **i.e A**
0		
1		

Idempotent Law:

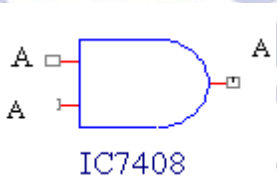
1. A+A = A



TRUTH TABLE:

INPUTS	OUTPUTS
A	A
0	
1	

2. A.A = A



TRUTH TABLE:

INPUTS	OUTPUTS
A	A
0	
1	

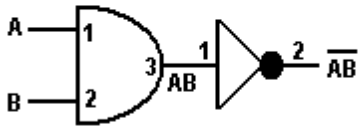
Demorgan's Law:

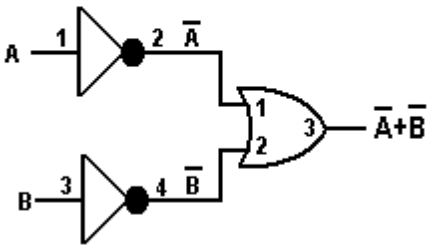
Law 1:

The complement of a product is equal to the sum of the complements.

LOGIC DIAGRAM:

$$\overline{AB} = \overline{A}+\overline{B}$$





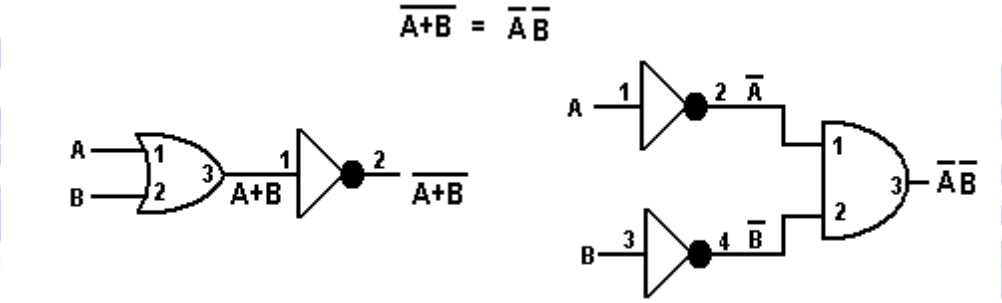
TRUTH TABLE:

INPUTS		OUTPUTS				
A	**B**	**AB**	$\overline{AB}$	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
0	**0**					
0	**1**					
1	**0**					
1	**1**					

Law 2:

The complement of a sum is equal to the product of the Complements.

LOGIC DIAGRAM:

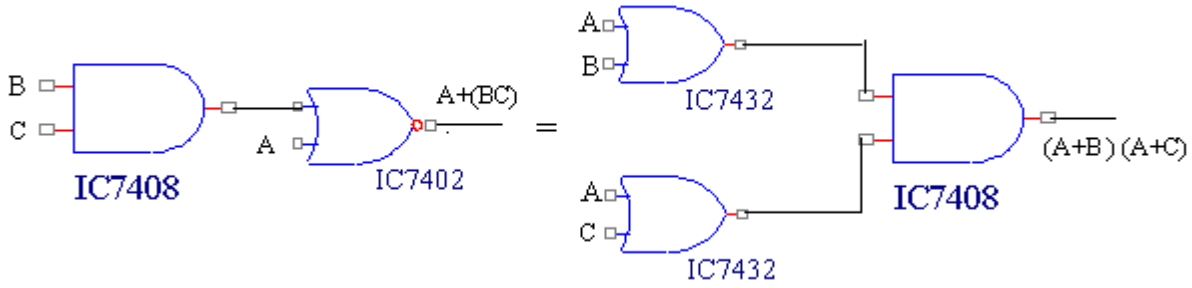


TRUTH TABLE:

INPUTS		OUTPUTS				
A	**B**	**A+B**	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$
0	**0**					
0	**1**					
1	**0**					
1	**1**					

Distributive Law:

$$A+(B.C) = (A+B).(A+C)$$



TRUTH TABLE:

INPUTS			OUTPUTS				
A	**B**	**C**	**B.C**	**A+(B.C)**	**A+B**	**A+C**	**(A+B).(A+C)**
0	**0**	**0**					
0	**0**	**1**					
0	**1**	**0**					
0	**1**	**1**					
1	**0**	**0**					
1	**0**	**1**					
1	**1**	**0**					
1	**1**	**1**					

PROCEDURE:

1. Circuit connections are given as per in the given circuit diagram.
2. Power supply is switched on.
3. The various set of inputs are given and the corresponding outputs are obtained.
4. The outputs are verified with the truth table.

RESULT:

Thus the truth tables for the Boolean function are verified.

VIVA QUESTIONS

EXP.NO: 3

DESIGN AND IMPLEMENTATION OF ADDERS OR SUBSTRACTOR USING LOGIC GATES

AIM:

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

APPARATUS REQUIRED:

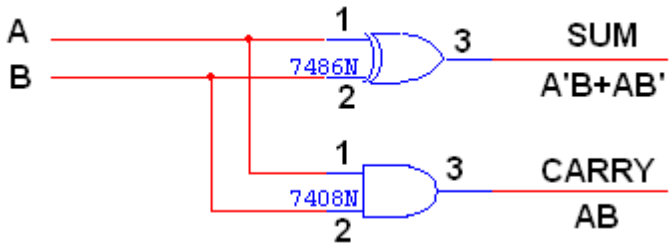
Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	AND GATE	IC 7408	1
2.	X-OR GATE	IC 7486	1
3.	NOT GATE	IC 7404	1
4.	OR GATE	IC 7432	1
3.	IC TRAINER KIT	-	1
4.	PATCH CORDS	-	Req.

PROCEDURE:

1. Circuit connections are given as per in the diagram.
2. Power supply is switched on.
3. All the set of inputs are given and the corresponding outputs are verified with truth table.
4. Same procedure is followed for all set laws.

HALF ADDER:

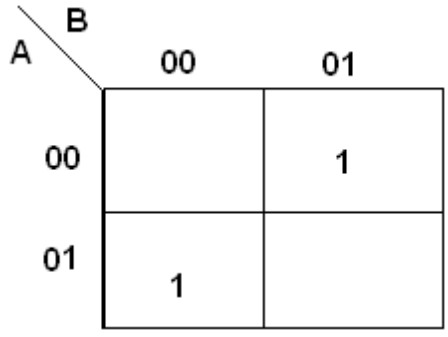
LOGIC DIAGRAM:



TRUTH TABLE:

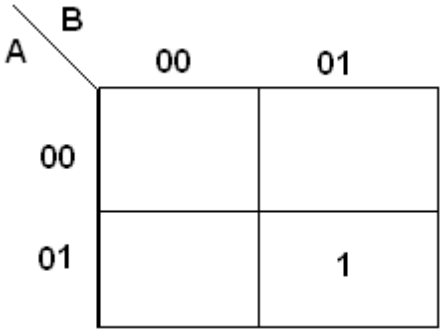
INPUTS		OUTPUTS	
A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

K-Map for SUM:



SUM = A'B + AB'

K-Map for CARRY:

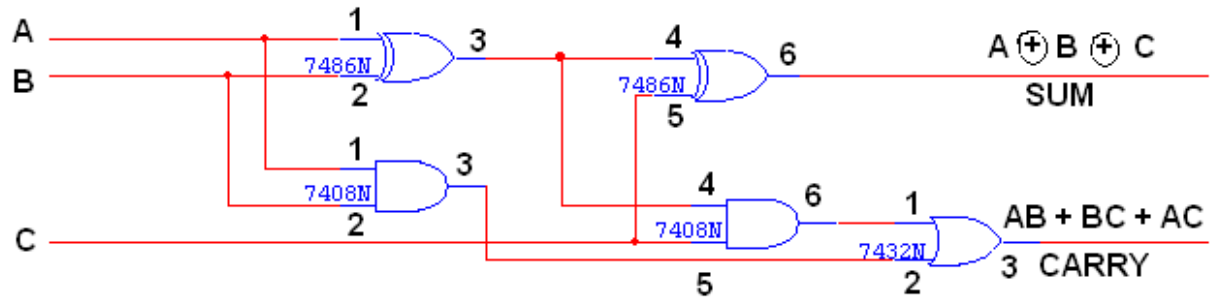


CARRY = AB

FULL ADDER:

LOGIC DIAGRAM:

FULL ADDER USING TWO HALF ADDER



TRUTH TABLE:

INPUTS			OUTPUTS	
A	B	C	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

K-Map for SUM:

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

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

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.

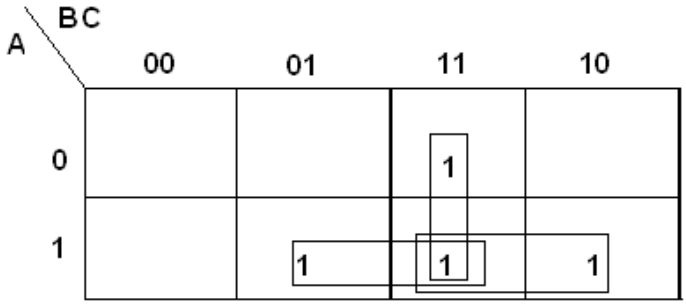
HALF SUBTRACTOR:

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.

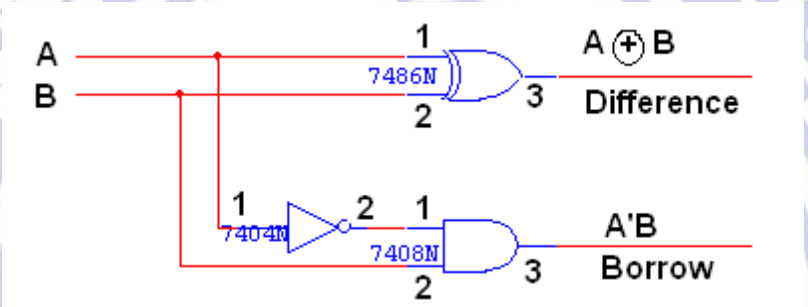
K-Map for CARRY:



CARRY = AB + BC + AC

HALF SUBTRACTOR

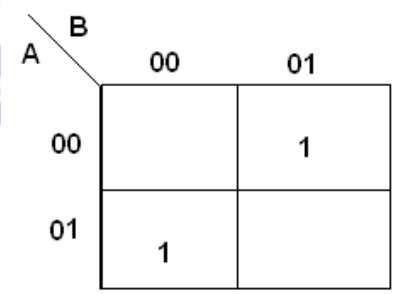
LOGIC DIAGRAM:



TRUTH TABLE:

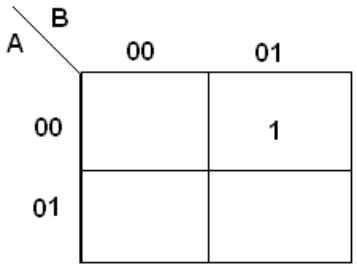
INPUTS		OUTPUTS	
A	**B**	**BORROW**	**DIFFERENCE**
0	**0**	**0**	**0**
0	**1**	**1**	**1**
1	**0**	**0**	**1**
1	**1**	**0**	**0**

K-Map for DIFFERENCE:



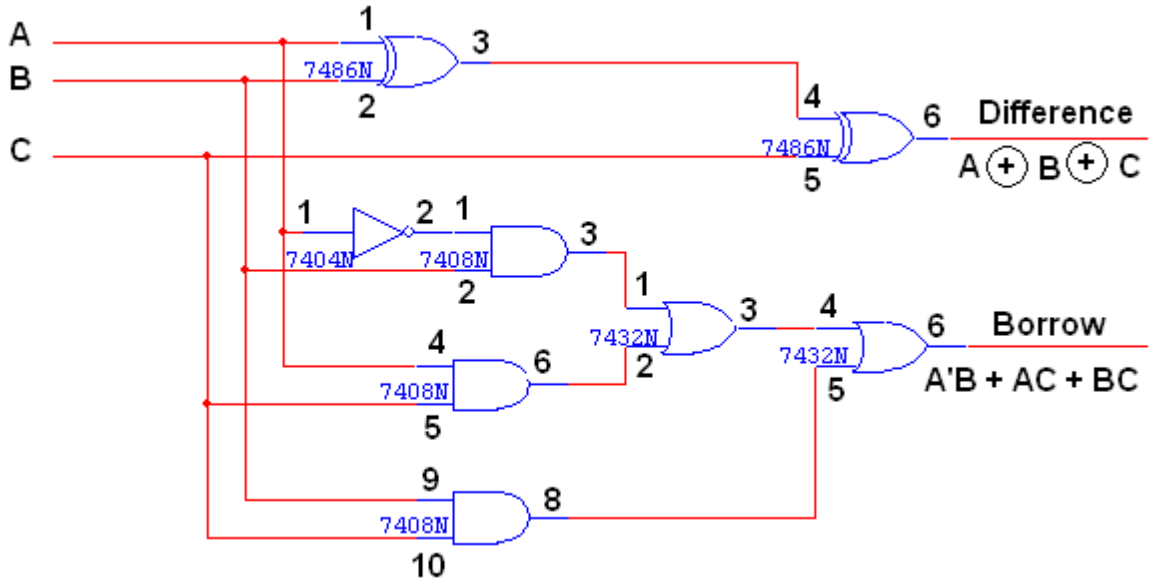
DIFFERENCE = A'B + AB'

K-Map for BORROW:

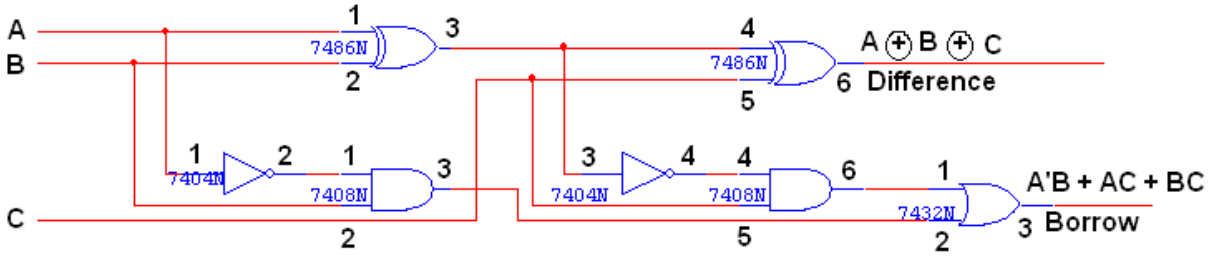


BORROW = A'B

FULL SUBTRACTOR LOGIC DIAGRAM:



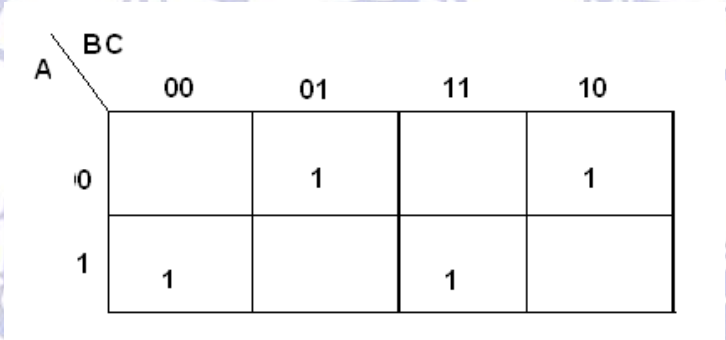
FULL SUBTRACTOR USING TWO HALF SUBTRACTOR:



TRUTH TABLE:

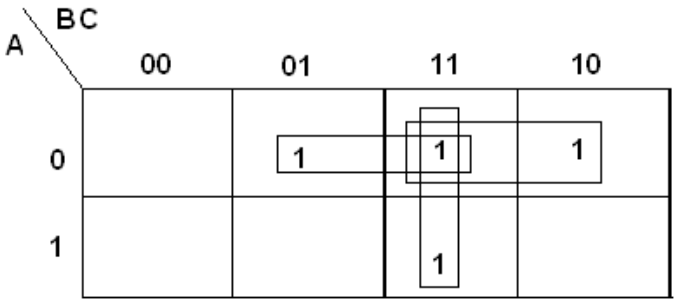
INPUTS			OUTPUTS	
A	B	C	BORROW	DIFFERENCE
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

K-Map for Difference:



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

K-Map for Borrow:



Borrow = A'B + BC + A'C

RESULT:

Thus the adder and substractor circuit are verified and design using truth table.

VIVA QUESTIONS

EXP NO: 4

DESIGN & IMPLEMENTATION OF CODE CONVERTOR

AIM:

To design and implement 4-bit

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

APPARATUS REQUIRED:

Sl. No.	COMPONENTS	SPECIFICATION/ RANGE	Qty
1.	X-OR GATE	IC 7486	2
2.	AND GATE	IC 7408	1
3.	NOT GATE	IC 7404	1
4.	OR GATE	IC 7432	1
5.	DIGITAL TRAINER KIT	-	1
6.	CONNECTING WIRES	-	Req.

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.

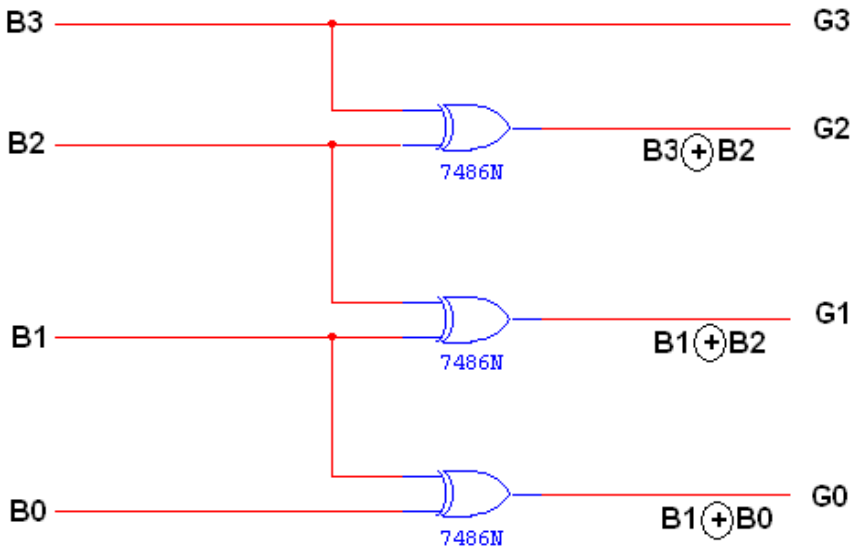
PROCEDURE:

(i) Connections were given as per circuit diagram.

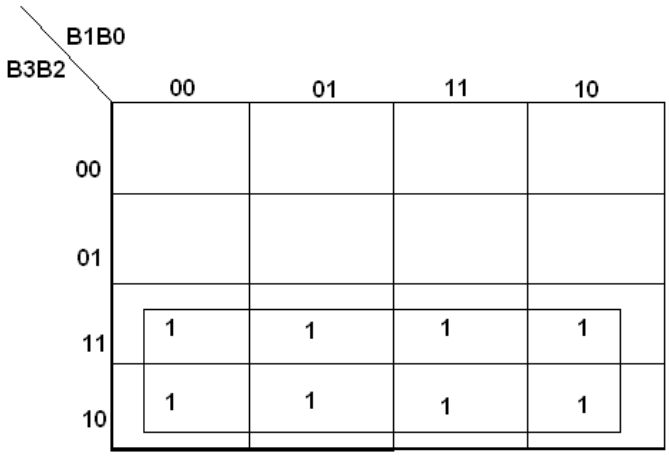
(ii) Logical inputs were given as per truth table

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

LOGIC DIAGRAM:
BINARY TO GRAY CODE CONVERTOR

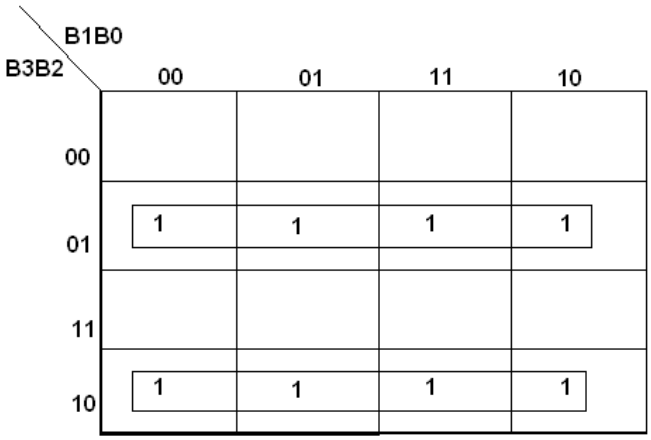


K-Map for G_3:



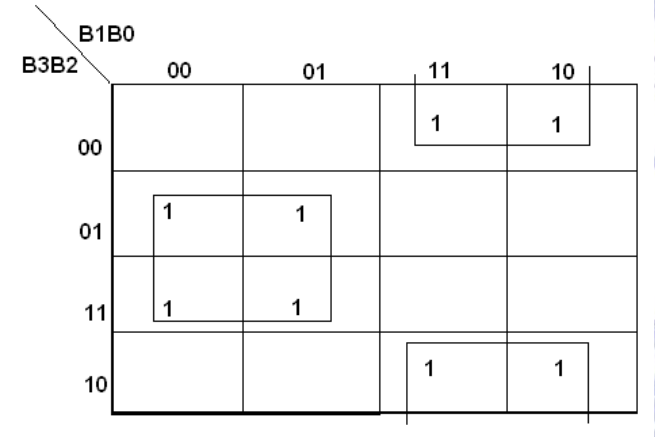
$G_3 = B_3$

K-Map for G_2:



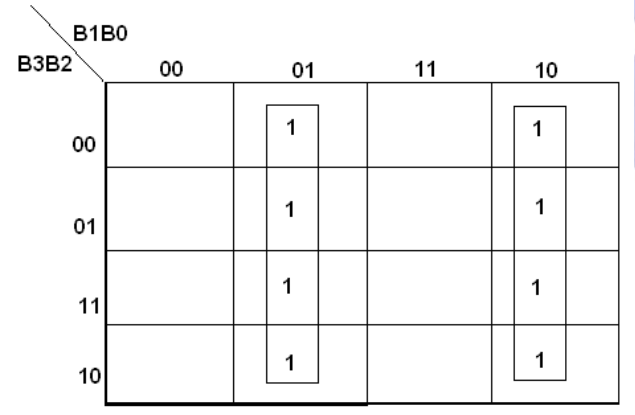
$G2 = B3 \oplus B2$

K-Map for G_1:



$G1 = B1 \oplus B2$

K-Map for G_0:



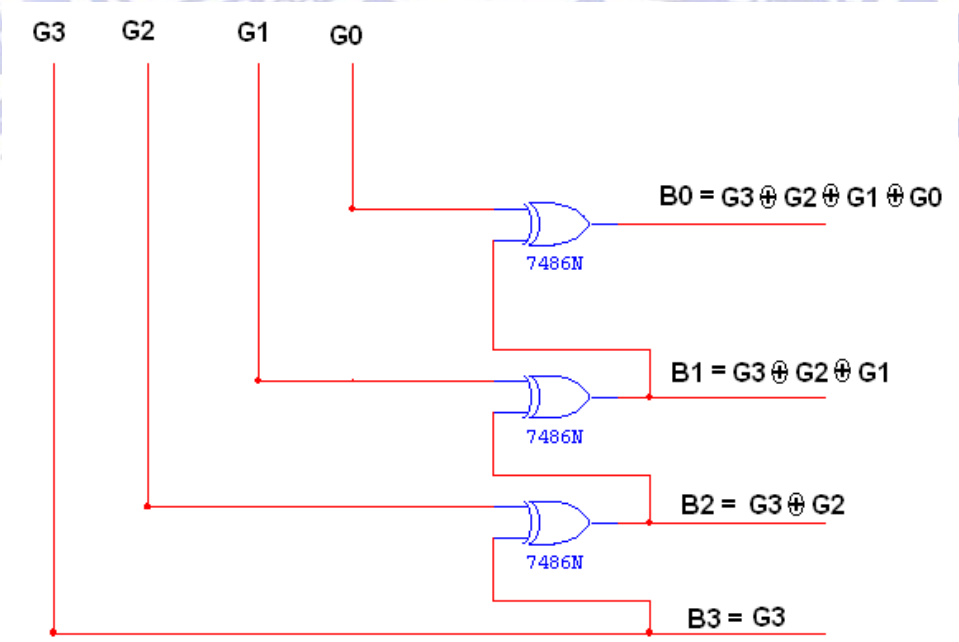
$G0 = B1 \oplus B0$

TRUTH TABLE:

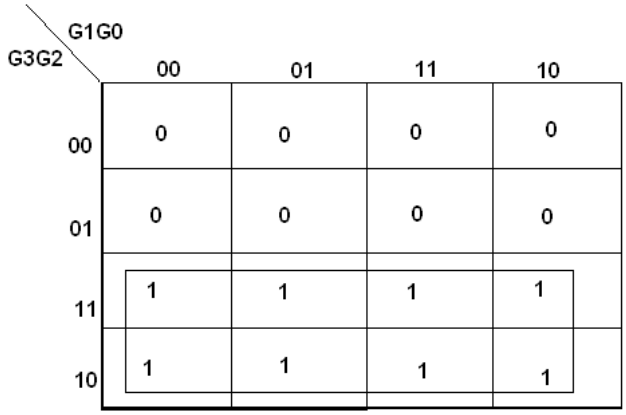
Binary input				Gray code output			
B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1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

LOGIC DIAGRAM:

GRAY CODE TO BINARY CONVERTOR

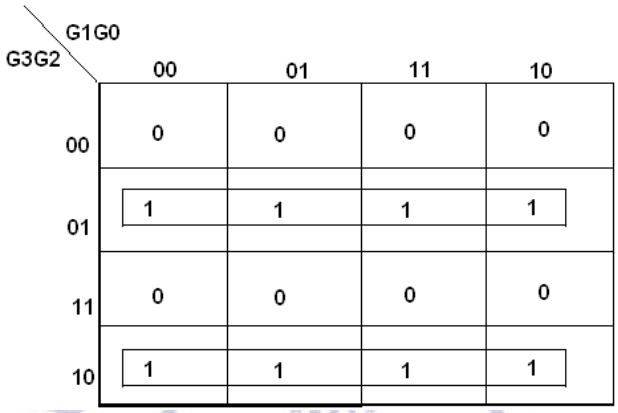


K-Map for B_3:



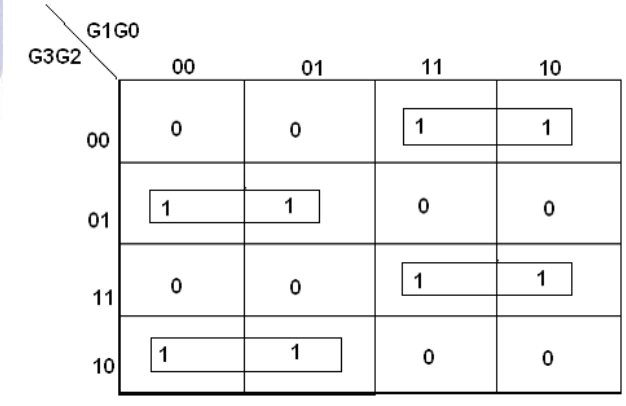
B3 = G3

K-Map for B_2:



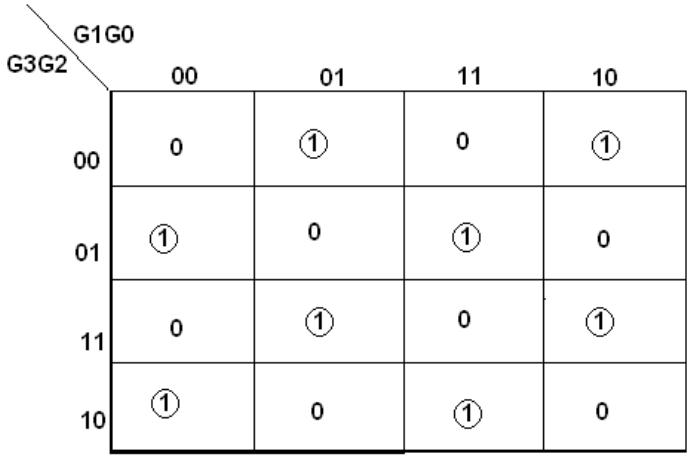
B2 = G3⊕G2

K-Map for B_1:



B1 = G3⊕G2⊕G1

K-Map for B_0:

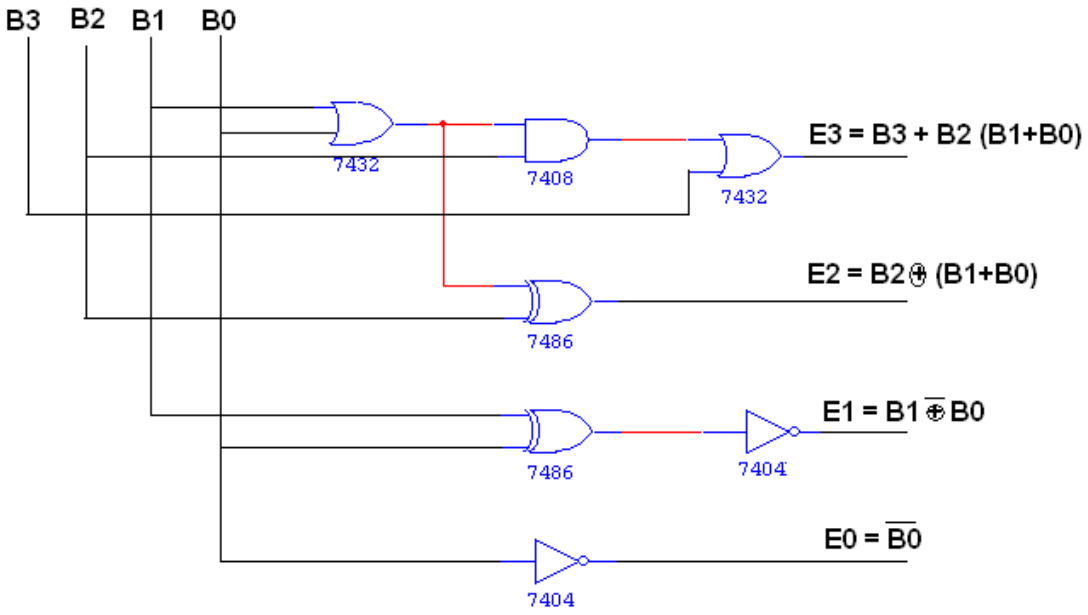


$B0 = G3 \oplus G2 \oplus G1 \oplus G0$

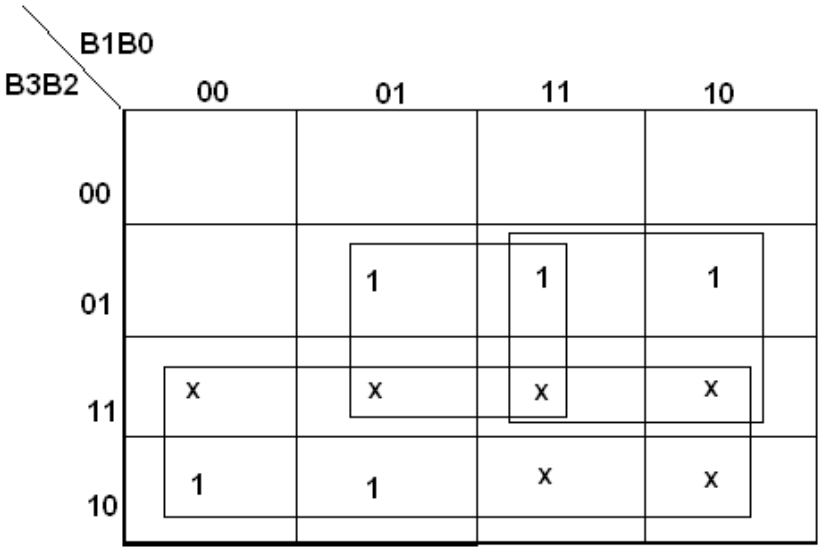
TRUTH TABLE:

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

LOGIC DIAGRAM:
BCD TO EXCESS-3 CONVERTOR

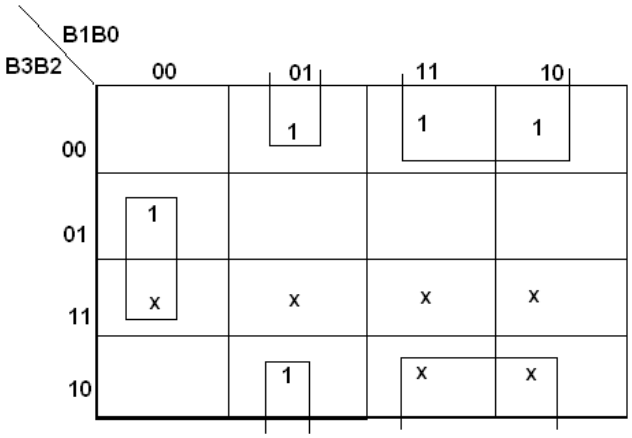


K-Map for E_3:



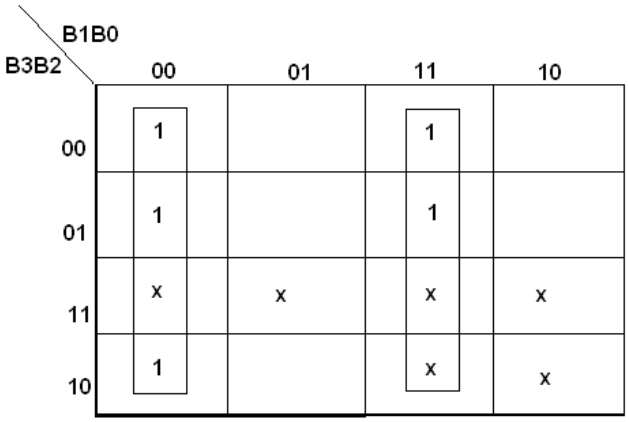
E3 = B3 + B2 (B0 + B1)

K-Map for E_2:



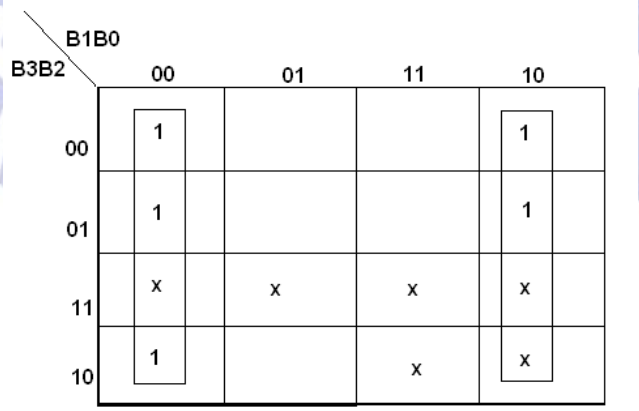
$$E2 = B2 \oplus (B1 + B0)$$

K-Map for E_1:



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

K-Map for E_0:

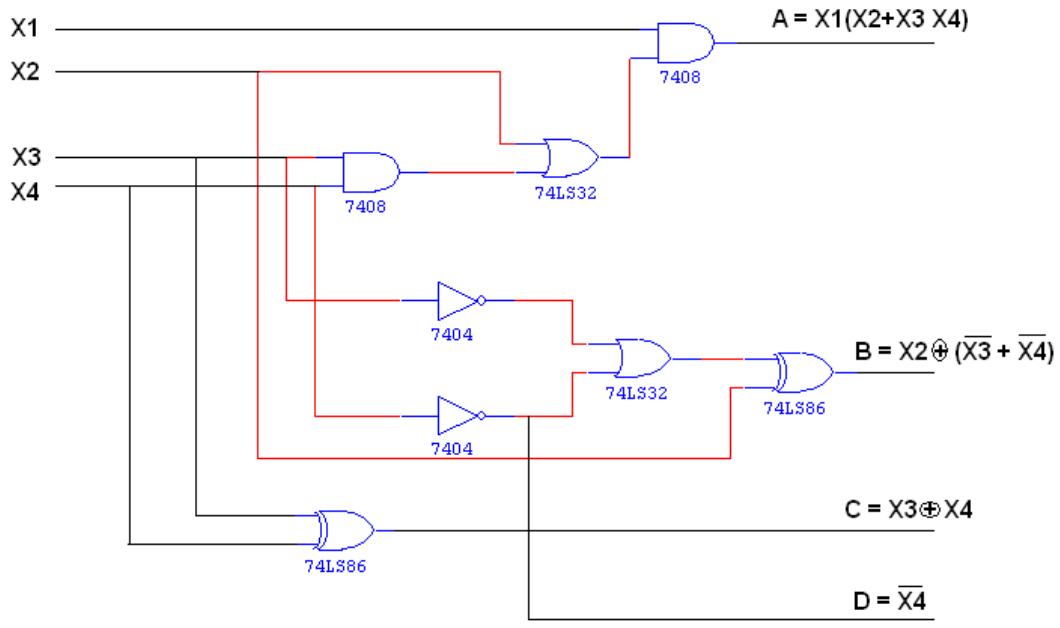


$$E0 = \overline{B0}$$

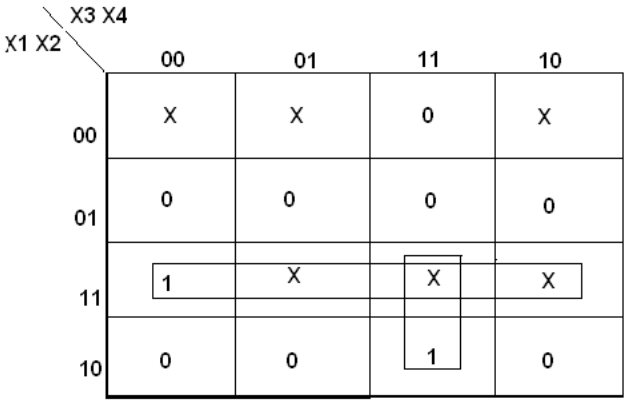
TRUTH TABLE:

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

LOGIC DIAGRAM:
EXCESS-3 TO BCD CONVERTOR

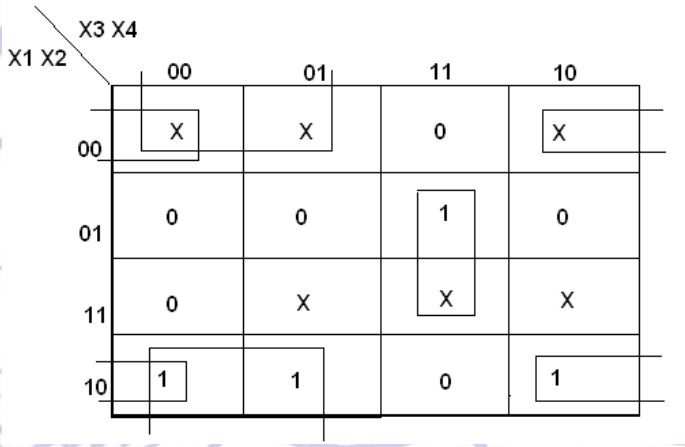


K-Map for A:



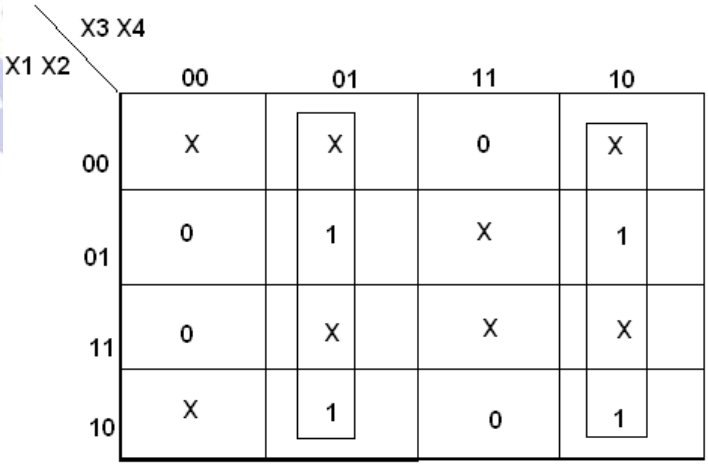
A = X1 X2 + X3 X4 X1

K-Map for B:



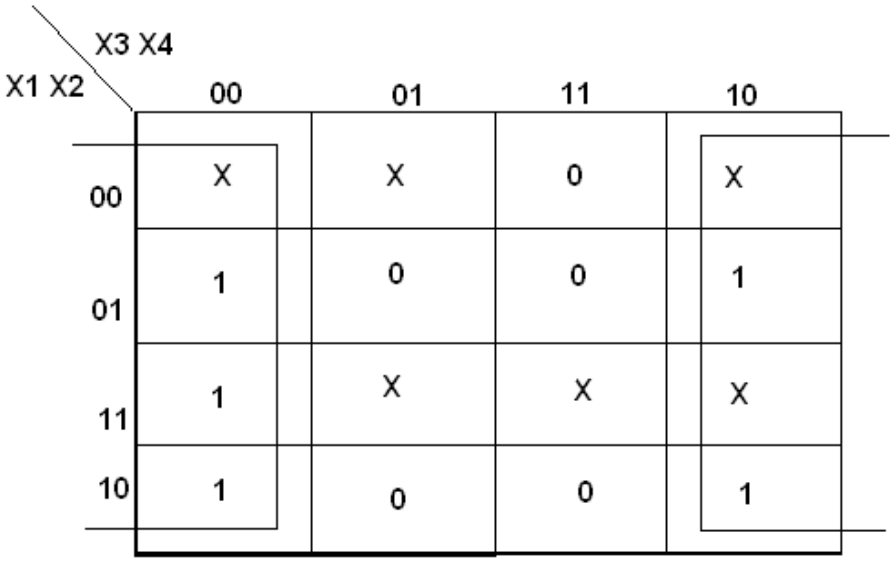
$$B = X2 \oplus (\overline{X3} + \overline{X4})$$

K-Map for C:



$$C = X3 \oplus X4$$

K-Map for D:



$D = \overline{X4}$

TRUTH TABLE:

Excess – 3 output				BCD input			
B3	**B2**	**B1**	**B0**	**B3**	**B2**	**B1**	**B0**
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

RESULT:

Thus the code converters were designed and verified using the corresponding truth table.

EXP NO: 5

DESIGN OF 4-BIT ADDER AND SUBTRACTOR

AIM:

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

APPARATUS REQUIRED:

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

THEORY:

4 BIT BINARY ADDER:

A binary adder is a digital circuit that produces the arithmetic sum of two binary numbers. It can be constructed with full adders connected in cascade, with the output carry from each full adder connected to the input carry of next full adder in chain. The augends bits of 'A' and the addend bits of 'B' are designated by subscript numbers from right to left, with subscript 0 denoting the least significant bits. The carries are connected in chain through the full adder. The input carry to the adder is C0 and it ripples through the full adder to the output carry C4.

4 BIT BINARY SUBTRACTOR:

The circuit for subtracting A-B consists of an adder with inverters, placed between each data input 'B' and the corresponding input of full adder. The input carry C0 must be equal to 1 when performing subtraction.

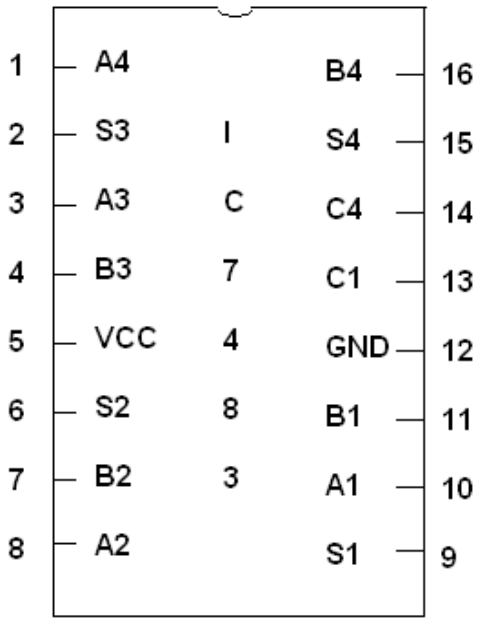
4 BIT BINARY ADDER/SUBTRACTOR:

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

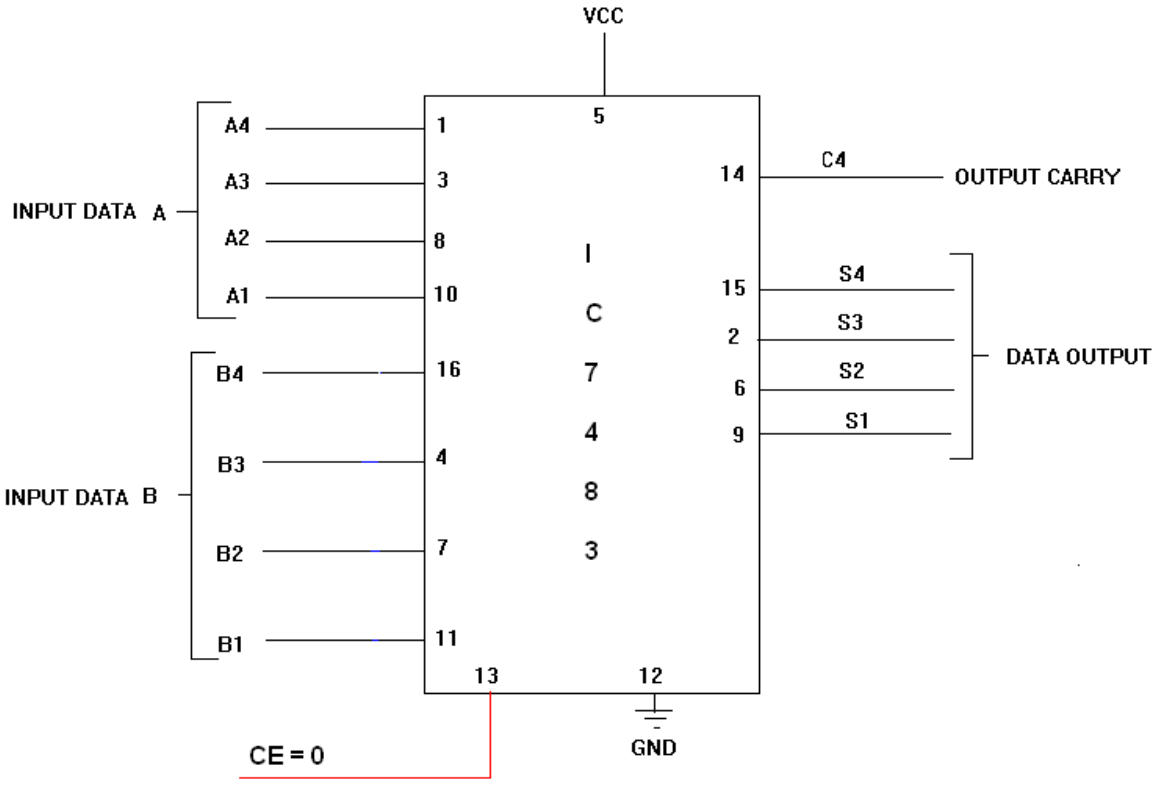
4 BIT BCD ADDER:

Consider the arithmetic addition of two decimal digits in BCD, together with an input carry from a previous stage. Since each input digit does not exceed 9, the output sum cannot be greater than 19, the 1 in the sum being an input carry. The output of two decimal digits must be represented in BCD and should appear in the form listed in the columns. ABCD adder that adds 2 BCD digits and produce a sum digit in BCD. The 2 decimal digits, together with the input carry, are first added in the top 4 bit adder to produce the binary sum.

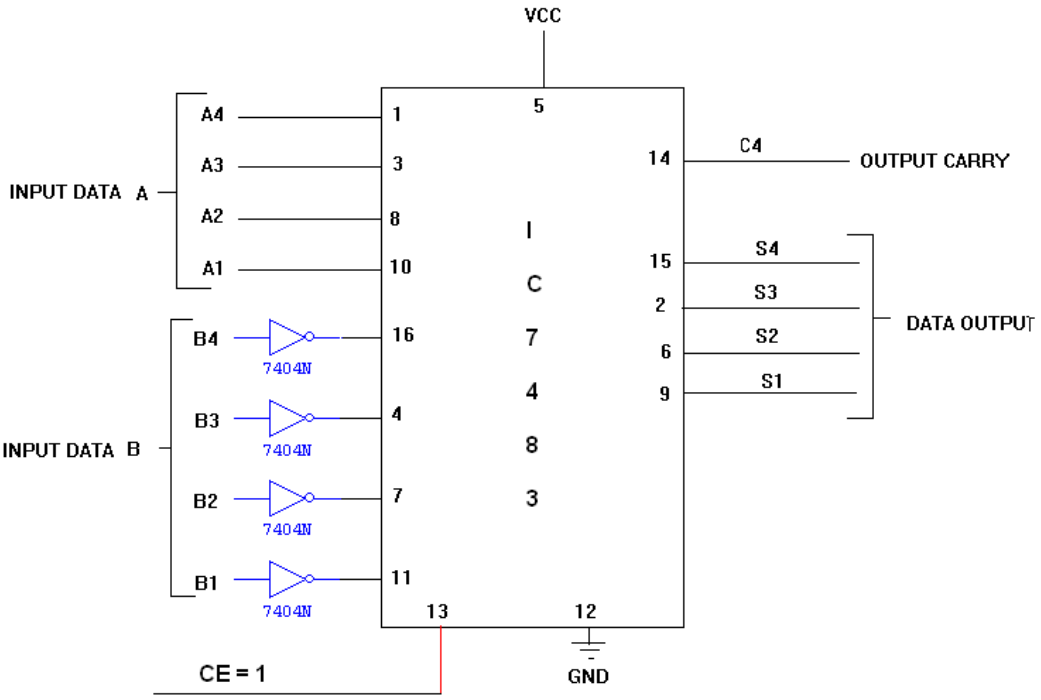
PIN DIAGRAM FOR IC 7483:



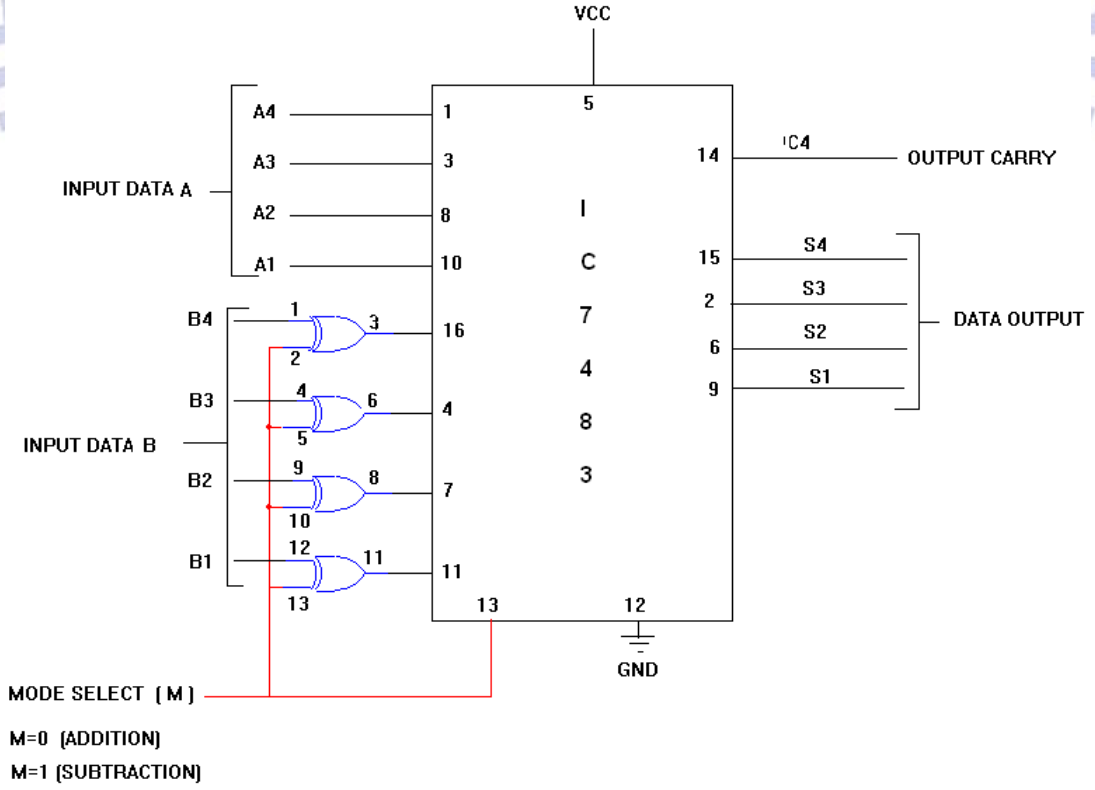
LOGIC DIAGRAM:
4-BIT BINARY ADDER



LOGIC DIAGRAM:
4-BIT BINARY SUBTRACTOR



LOGIC DIAGRAM: 4-BIT BINARY ADDER/SUBTRACTOR



TRUTH TABLE:

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

RESULT:

Thus the 4-BIT adder/subtractor using IC 7483 was implemented and output was verified with the help of truth table.

EXP NO: 6 (a) ODD PARITY CHECKER /GENERATOR

AIM:

To design and verify the truth table of a three bit Odd Parity generator and checker.

APPARATUS REQUIRED:

S.No	Name of the Apparatus	Range	Quantity
1.	Digital IC trainer kit	-	1
2.	EX-OR gate	IC 7486	
3.	NOT gate	IC 7404	
4.	Connecting wires		required

THEORY:

A parity bit is used for the purpose of detecting errors during transmission of binary information. A parity bit is an extra bit included with a binary message to make the number of 1's either odd or even. The message including the parity bit is transmitted and then checked at the receiving end for errors.

An error is detected if the checked parity does not correspond with the one transmitted. The circuit that generates the parity bit in the transmitter is called a parity generator 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 of 1's an even amount and in odd parity the added parity bit will make the total number of 1's an odd amount. In a three bit odd parity generator the three bits in the message together with the parity bit are transmitted to their destination, where they are applied to the parity checker circuit. The parity checker circuit checks for possible errors in the transmission.

Since the information was transmitted with odd parity the four bits received must have an odd number of 1's. An error occurs during the transmission if the four bits received have an even number of 1's, indicating that one bit has changed during transmission. The output of the parity checker is denoted by PEC (parity error check) and it will be equal to 1 if an error occurs, i.e., if the four bits received has an even number of 1's.

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

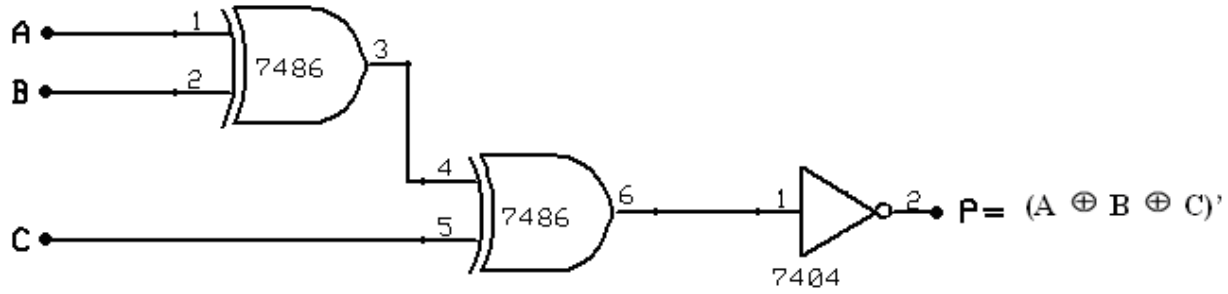
From the truth table the expression for the output parity bit is,

$$P(A, B, C) = \Sigma (0, 3, 5, 6)$$

Also written as, (Reduce using K-Map)

$$P = A'B'C' + A'BC + AB'C + ABC' = (A \oplus B \oplus C)'$$

CIRCUIT DIAGRAM:
ODD PARITY GENERATOR



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

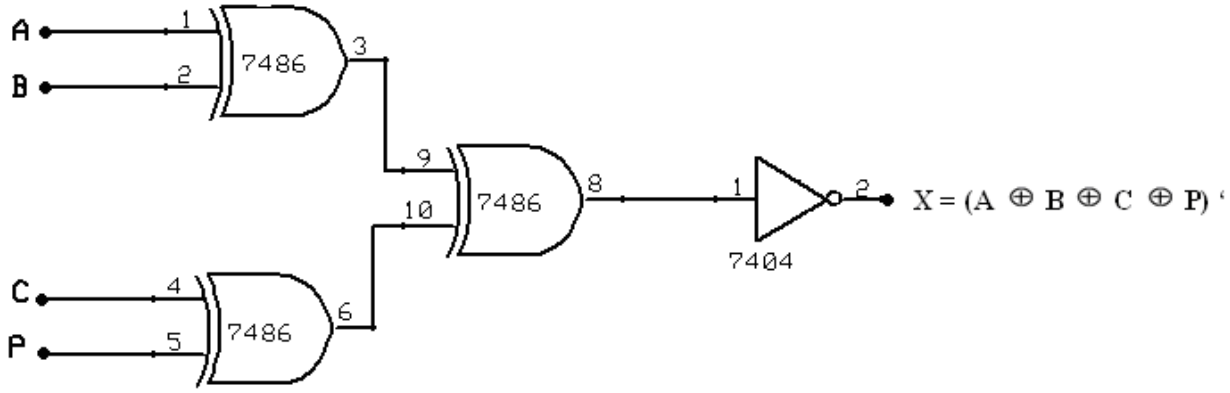
From the truth table the expression for the output parity checker bit is,

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

The above expression is reduced as,

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

CIRCUIT DIAGRAM:
ODD PARITY CHECKER



PROCEDURE:

1. Connections are given as per the circuit diagrams.
2. For all the ICs 7th pin is grounded and 14th pin is given +5 V supply.
3. Apply the inputs and verify the truth table for the Parity generator and checker.

RESULT:

The design of the three bit odd Parity generator and checker circuits was done and their truth tables were verified.

EXP NO: 6 (b) EVEN PARITY CHECKER /GENERATOR

AIM:

To design and verify the truth table of a three bit Even Parity generator and checker.

APPARATUS REQUIRED:

S.No	Name of the Apparatus	Range	Quantity
1.	Digital IC trainer kit	-	1
2.	EX-OR gate	IC 7486	
3.	Connecting wires		required

THEORY:

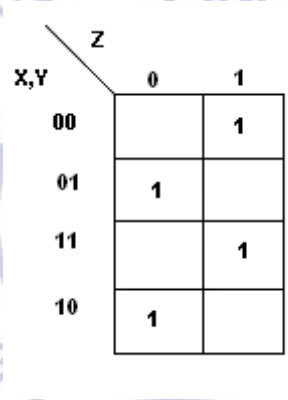
An even parity bit generator generates an output of 0 if the number of 1's in the input sequence is even and 1 if the number of 1's in the input sequence is odd. The checker circuit gives an output of 0 if there is no error in the parity bit generated. Thus it basically checks to see if the parity bit generator is error free or not.

EVEN PARITY GENERATOR
TRUTH TABLE:

S.No	INPUT (Three bit message)			OUTPUT (Even Parity bit)
	A	B	C	P
1.	0	0	0	0
2.	0	0	1	1
3.	0	1	0	1
4.	0	1	1	0

5.	**1**	**0**	**0**	**1**
6.	**1**	**0**	**1**	**0**
7.	**1**	**1**	**0**	**0**
8.	**1**	**1**	**1**	**1**

The circuit can now be derived by drawing the K-map for the output.



From this the minimal output equation is

$$P = \overline{X}\,\overline{Y}Z + \overline{X}Y\overline{Z} + XYZ + X\overline{Y}\,\overline{Z} = X \oplus Y \oplus Z$$

This function can be implemented using exclusive-or gates. The schematic of the parity generator circuit is shown in Figure 1.

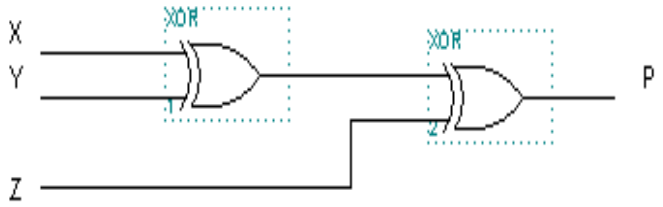


Figure 1: Parity bit generator

Similarly the checker circuit can be designed using XOR gates, where $C = X \oplus Y \oplus Z \oplus P$ and the circuit is shown in Figure 2.

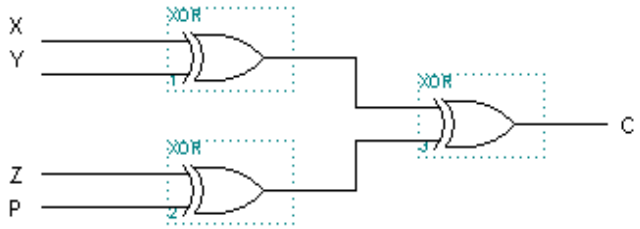


Figure 2: Checker circuit

EVEN PARITY CHECKER
TRUTH TABLE:

Message X Y Z	Even parity bit P	Checker bit C
0 0 0	0	0
0 0 1	1	0
0 1 0	1	0
0 1 1	0	0
1 0 0	1	0
1 0 1	0	0
1 1 0	0	0
1 1 1	1	0

PROCEDURE:

1. Connections are given as per the circuit diagrams.
2. For all the ICs 7th pin is grounded and 14th pin is given +5 V supply.
3. Apply the inputs and verify the truth table for the Parity generator and checker.

RESULT:

The design of the three bit even Parity generator and checker circuits was done and their truth tables were verified.

EXP NO: 7 DESIGN AND IMPLEMENTATION OF MAGNITUDE COMPARATOR

AIM:

To design and implement

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

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

APPARATUS REQUIRED:

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

THEORY:

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

A = A3 A2 A1 A0

B = B3 B2 B1 B0

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

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

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

$A>B = A3B3^1 + X3A2B2^1 + X3X2A1B1^1 + X3X2X1A0B0^1$

$A<B = A3^1B3 + X3A2^1B2 + X3X2A1^1B1 + X3X2X1A0^1B0$

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

A = B is expanded as,

$$A = B = (A3 + B3)\ (A2 + B2)\ (A1 + B1)\ (A0 + B0)$$

x3 x2 x1 x0

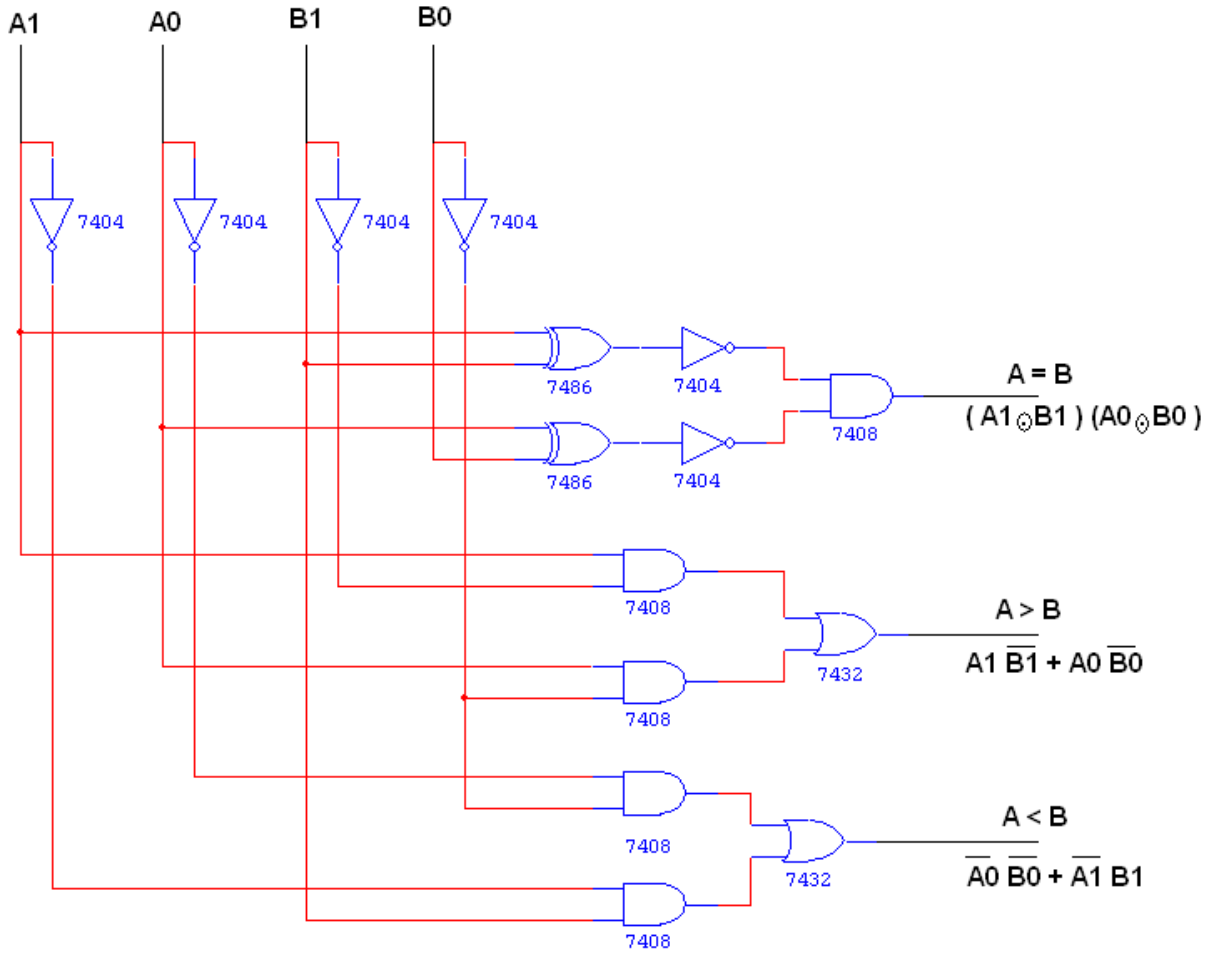
PROCEDURE:

(i) Connections are given as per circuit diagram.

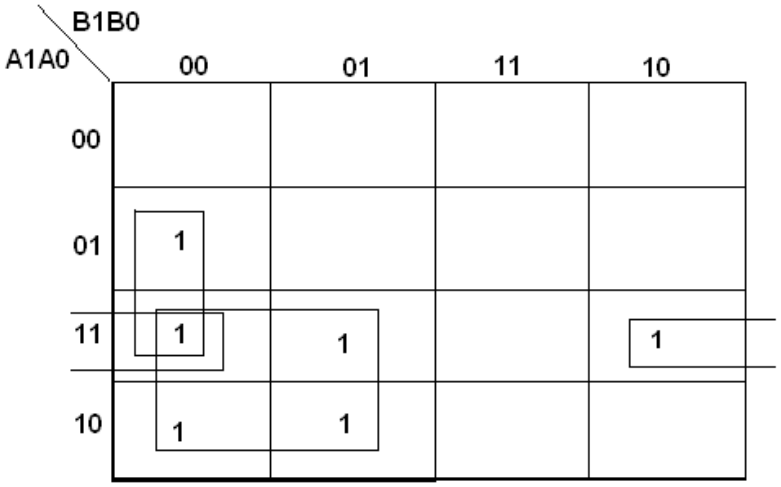
(ii) Logical inputs are given as per circuit diagram.

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

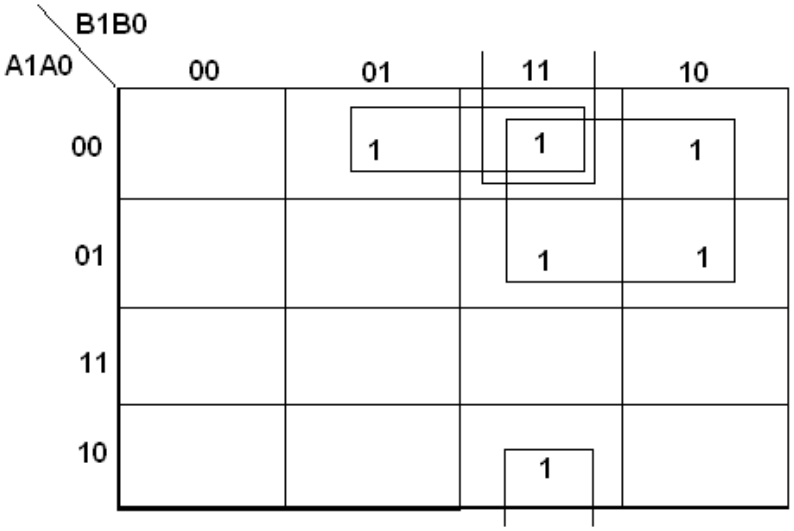
LOGIC DIAGRAM:
2 BIT MAGNITUDE COMPARATOR



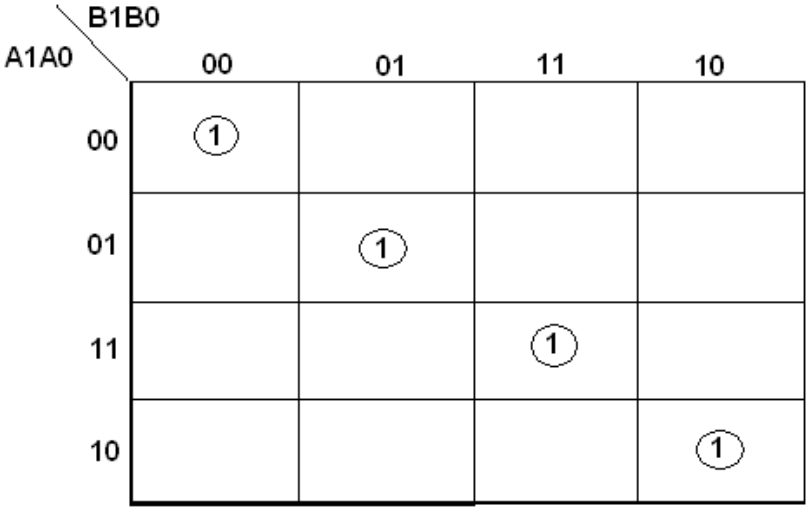
K MAP



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



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

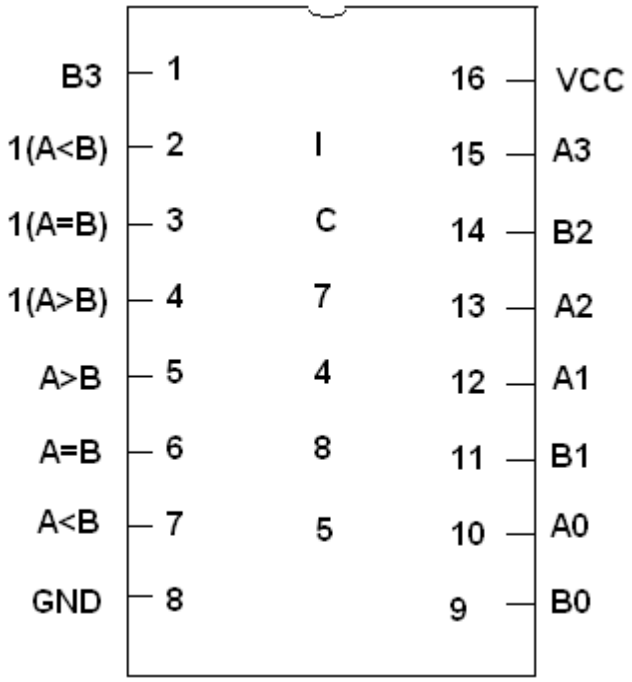


$A = B = (A0 \odot B0)(A1 \odot B1)$

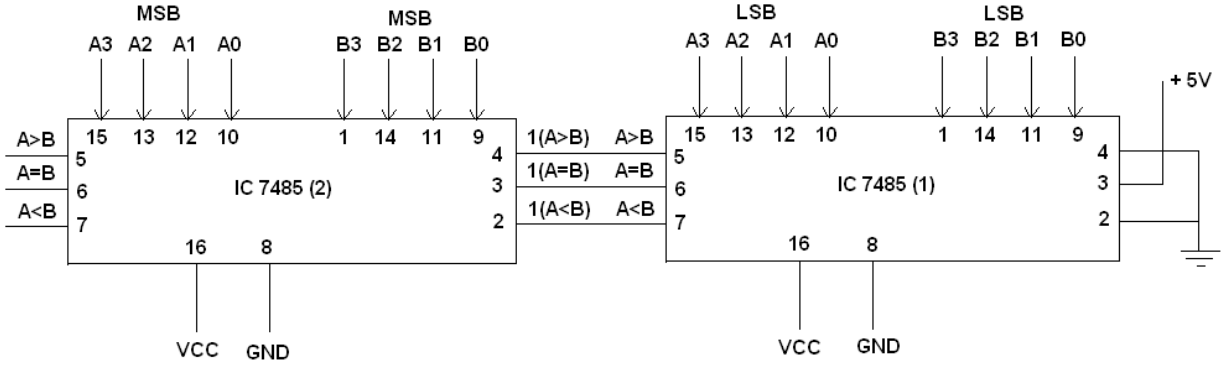
TRUTH TABLE

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

PIN DIAGRAM FOR IC 7485:



LOGIC DIAGRAM:
8 BIT MAGNITUDE COMPARATOR



TRUTH TABLE:

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

RESULT:

Thus the 2-bit and 8-bit magnitude comparator was designed and verified using the logic gates.

EXP NO: 8 DESIGN AND IMPLEMENTATION OF MULTIPLEXER AND DEMULTIPLEXER

AIM:

To design and implement Multiplexer and Demultiplexer using logic gates.

APPARATUS REQUIRED:

Sl. No.	COMPONENTS	SPECIFICATION/ RANGE	Qty
1.	3 I/P AND GATE (or) 2I/p	IC 7411 / IC 7408	2
2.	OR GATE	IC 7432	1
3.	NOT GATE	IC 7404	1
4.	DIGITAL TRAINER KIT	-	1
5.	CONNECTING WIRES	-	Req.

THEORY:

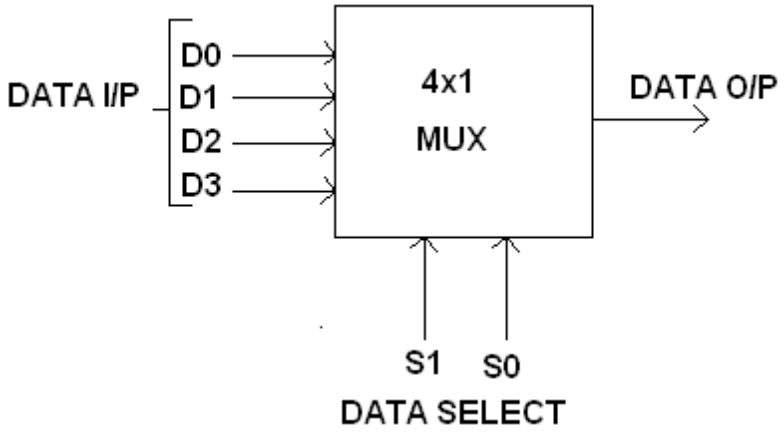
MULTIPLEXER:

Multiplexer means transmitting a large number of information units over a smaller number of channels or lines. A digital multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line. The selection of a particular input line is controlled by a set of selection lines. Normally there are 2^n input line and n selection lines whose bit combination determine which input is selected.

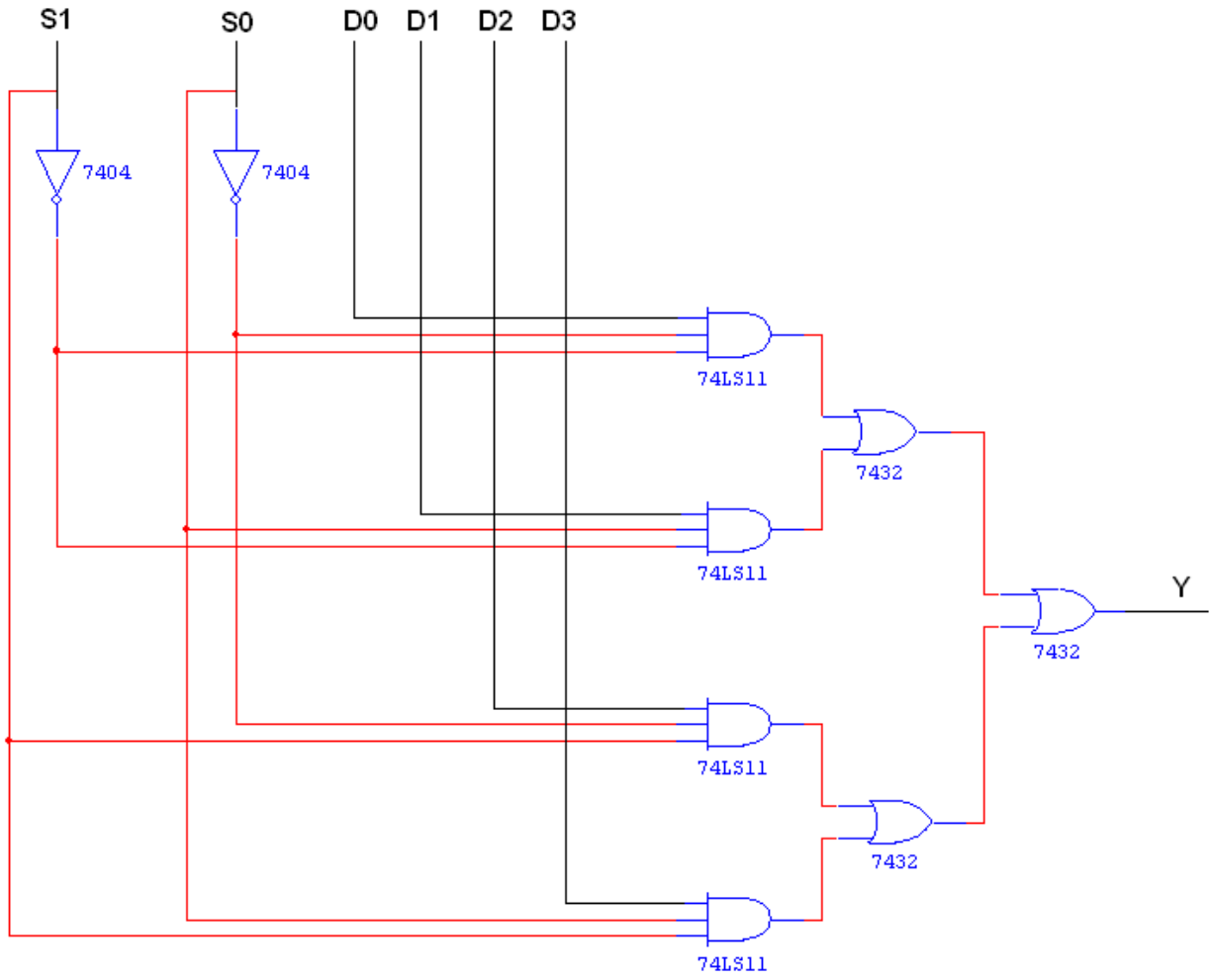
DEMULTIPLEXER:

The function of Demultiplexer is in contrast to multiplexer function. It takes information from one line and distributes it to a given number of output lines. For this reason, the demultiplexer is also known as a data distributor. Decoder can also be used as demultiplexer. In the 1: 4 demultiplexer circuit, the data input line goes to all of the AND gates. The data select lines enable only one gate at a time and the data on the data input line will pass through the selected gate to the associated data output line.

BLOCK DIAGRAM FOR 4:1 MULTIPLEXER:



CIRCUIT DIAGRAM FOR MULTIPLEXER:



FUNCTION TABLE:

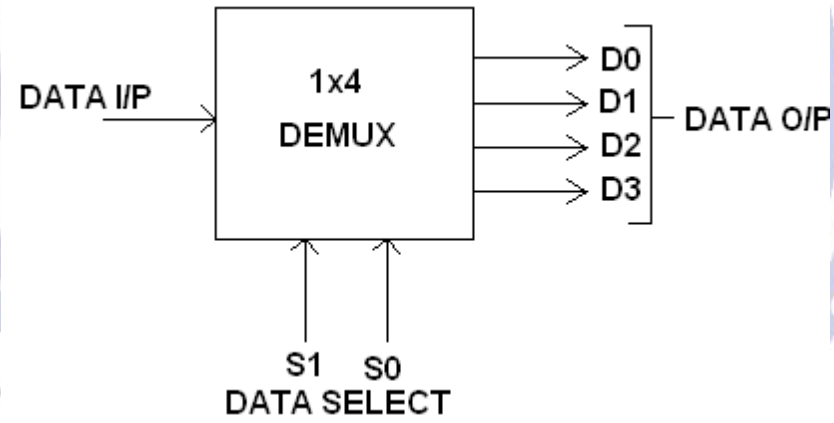
S1	S0	INPUTS (Y)-output
0	0	D0 → D0 S1' S0'
0	1	D1 → D1 S1' S0
1	0	D2 → D2 S1 S0'
1	1	D3 → D3 S1 S0

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

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

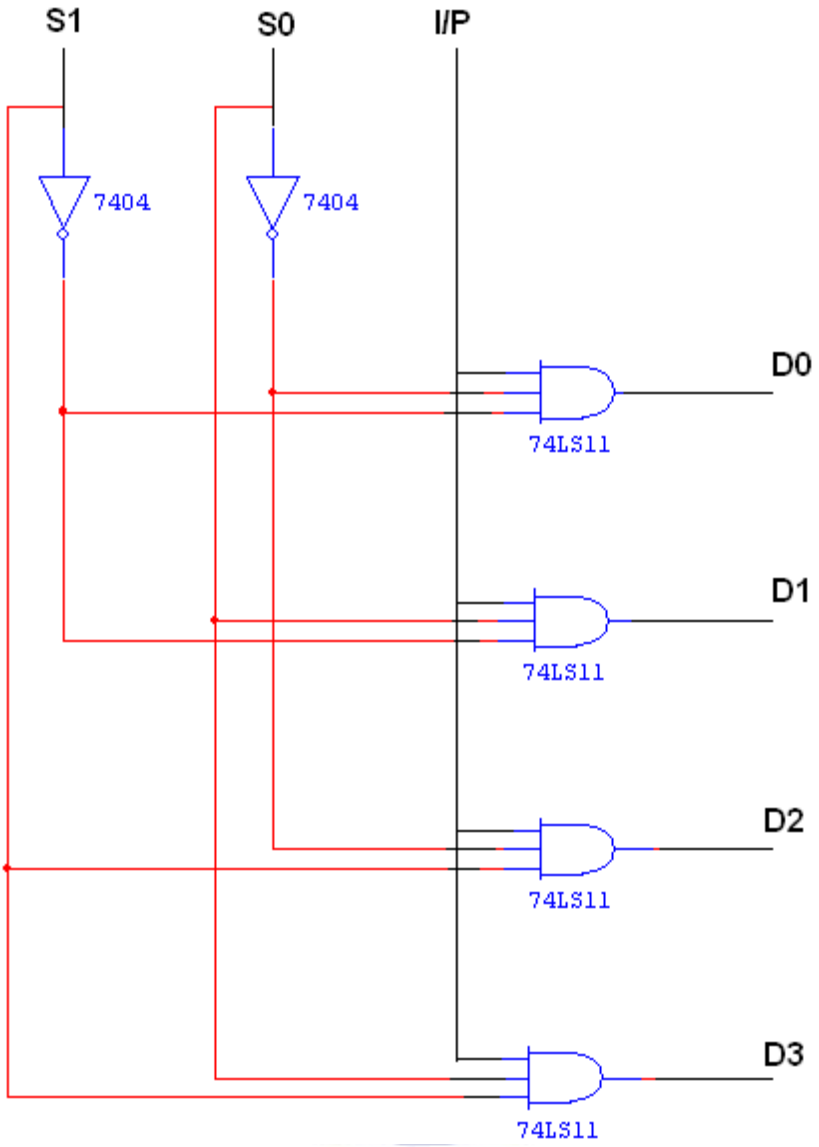
PROCEDURE:

(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.

LOGIC DIAGRAM FOR DEMULTIPLEXER:



TRUTH TABLE:

INPUT			OUTPUT			
I/P	**S1**	**S0**	**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

Result:

Thus the 4:1 Multiplexer and 1:4 Demultiplexer were designed using logic gates and truth table were verified.

EXP NO: 09 DESIGN AND IMPLEMENTATION OF SHIFT REGISTER

AIM:

To design and implement

(i) Serial in serial out

(ii) Serial in parallel out

(iii) Parallel in serial out

(iv) Parallel in parallel out

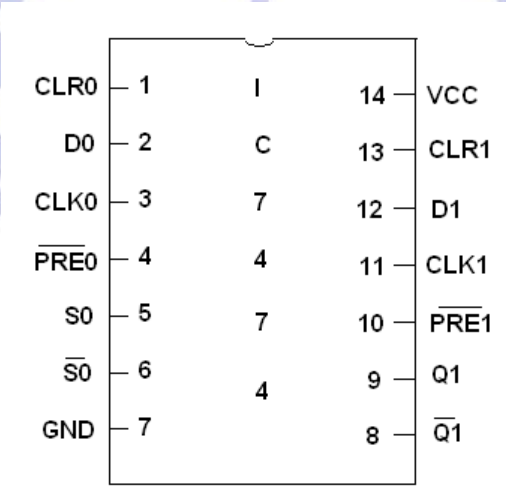
APPARATUS REQUIRED:

Sl. No.	COMPONENTS	SPECIFICATION/ RANGE	Qty
1.	D- FLIP FLOP	IC 7474	2
2.	OR GATE	IC 7432	1
3.	DIGITAL TRAINER KIT	-	1
4.	CONNECTING WIRES	-	Req.

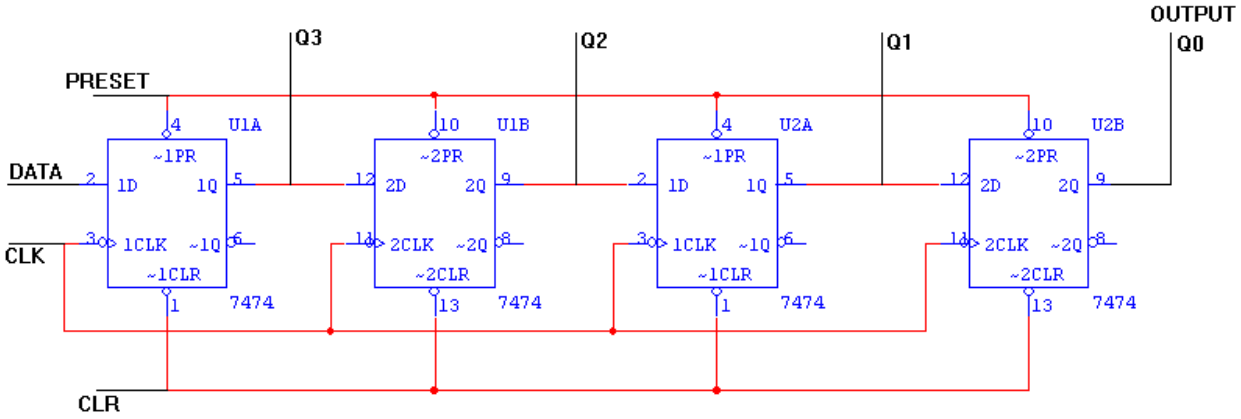
THEORY:

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:



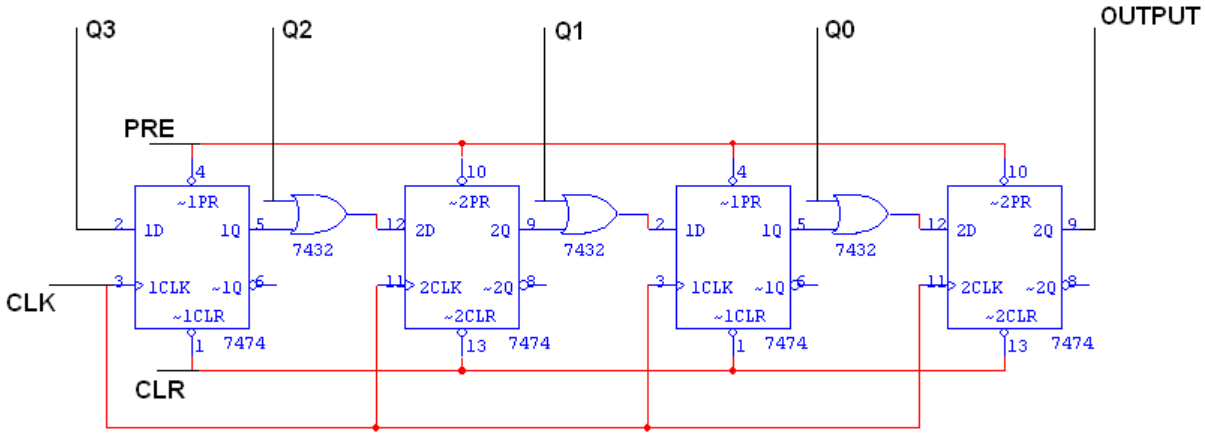
LOGIC DIAGRAM:
SERIAL IN PARALLEL OUT:



TRUTH TABLE:

CLK	DATA	OUTPUT			
		QA	QB	QC	QD
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:



PROCEDURE:

(i) Connections are given as per circuit diagram.

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

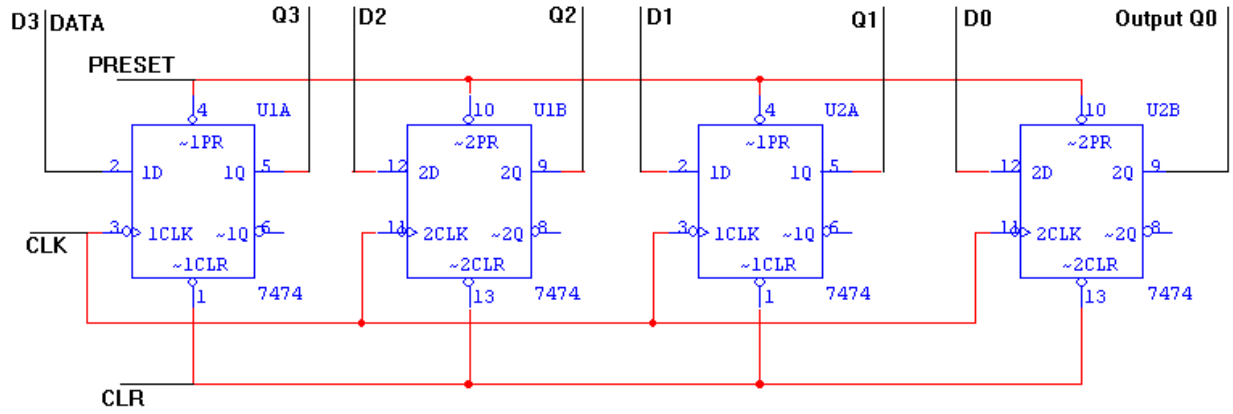
(iii) Observe the output and verify the truth table.

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:

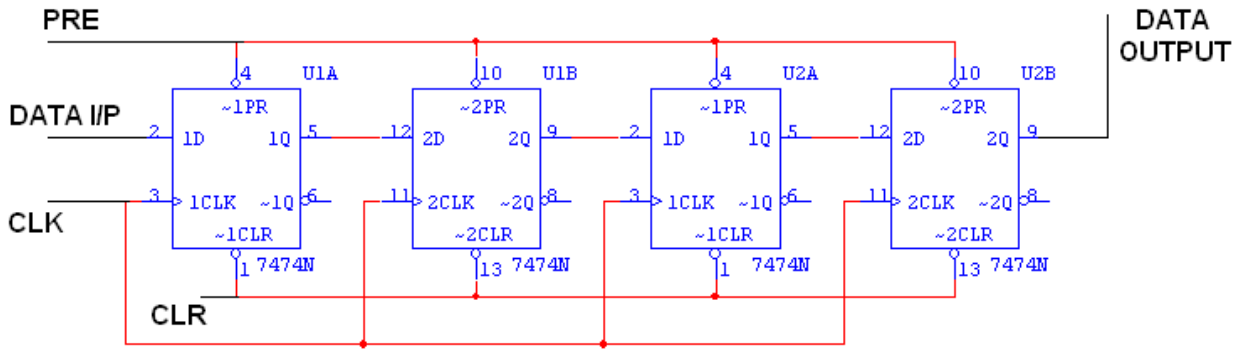
PARALLEL IN PARALLEL OUT:



TRUTH TABLE:

CLK	DATA INPUT				OUTPUT			
	DA	DB	DC	DD	QA	QB	QC	QD
1	1	0	0	1	1	0	0	1
2	1	0	1	0	1	0	1	0

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

RESULT:

Thus the implementation of shift registers using flip flops was completed successfully.

EXP NO: 10 DESIGN AND IMPLEMENTATION OF ASYNCHRONOUS / RIPPLE COUNTER USING IC 7476

AIM:

To design and implementation of 4 bit Asynchronous/Ripple counter, MOD 10 and MOD 12 Asynchronous/Ripple counter.

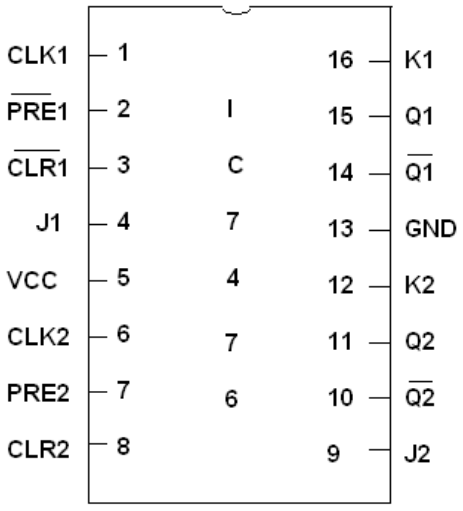
APPARATUS REQUIRED:

Sl. No.	COMPONENTS	SPECIFICATION/ RANGE	Qty
1.	JK FLIP FLOP	IC 7476	2
2.	NAND GATE	IC 7400	1
3.	DIGITAL TRAINER KIT	-	1
4.	CONNECTING WIRES	-	Req.

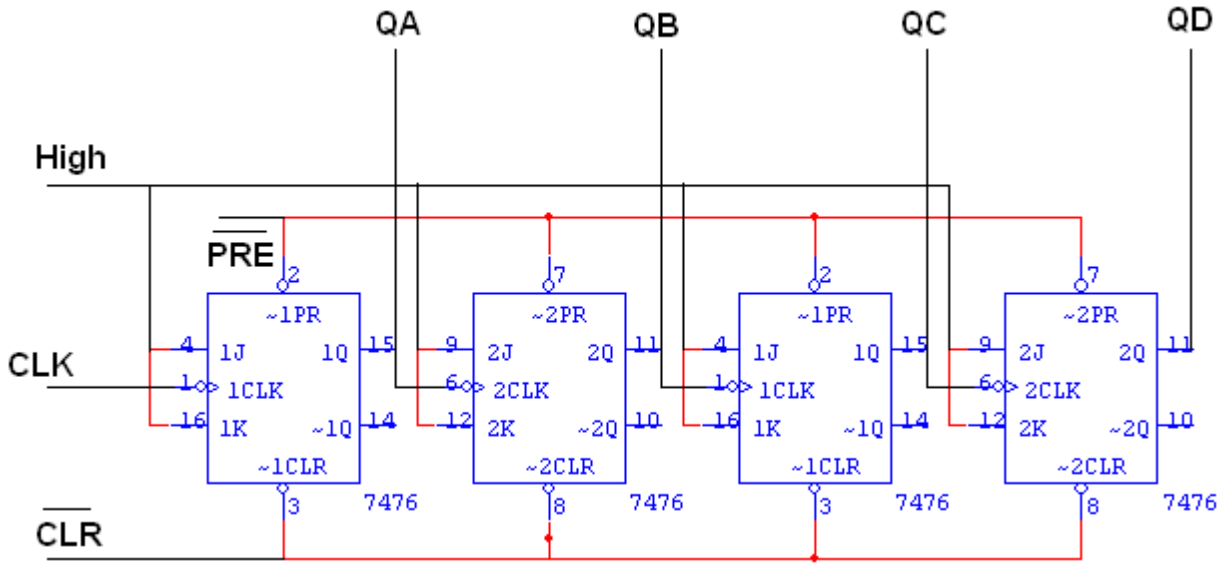
THEORY:

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:

CLOCK	OUTPUT			
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

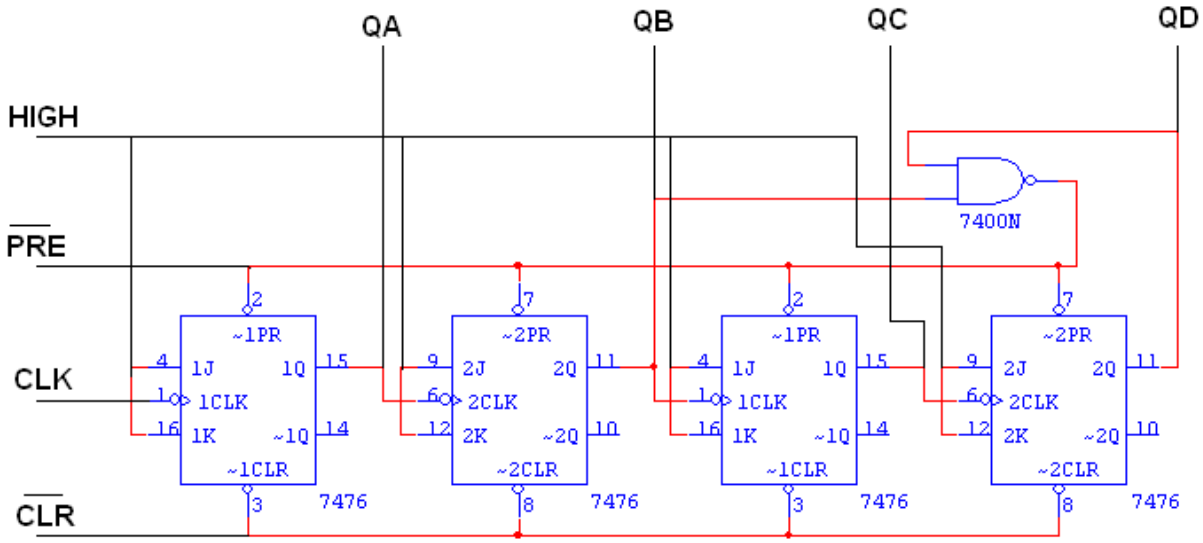
PROCEDURE:

(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.

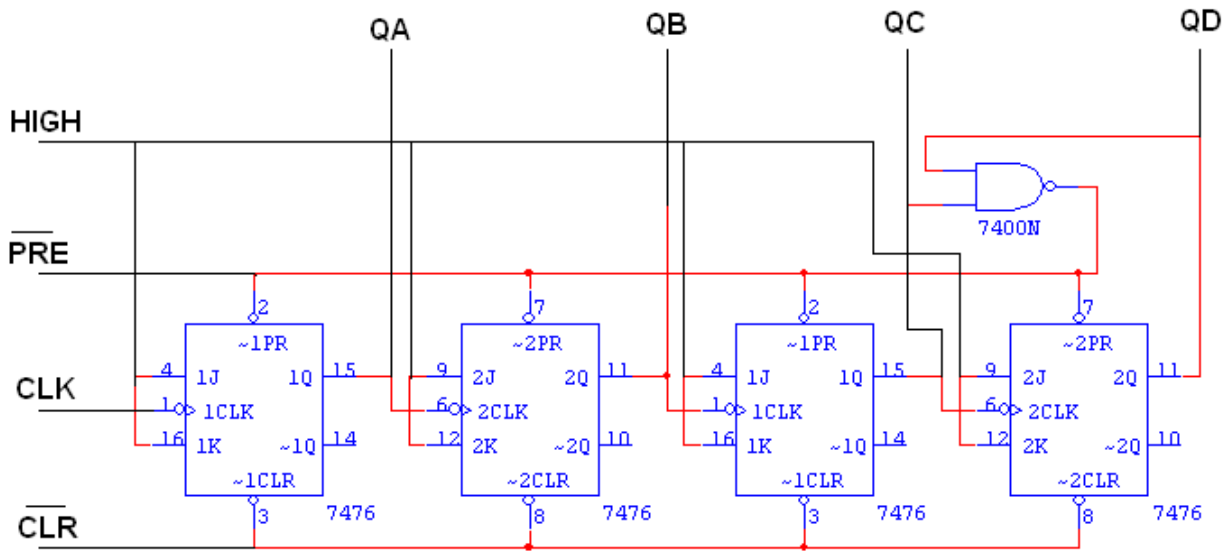
LOGIC DIAGRAM FOR MOD - 10 RIPPLE COUNTER:



TRUTH TABLE:

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

LOGIC DIAGRAM FOR MOD - 12 RIPPLE COUNTER:



TRUTH TABLE:

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

Result:

Thus the 4 bit Asynchronous/Ripple counter, MOD 10 and MOD 12 Asynchronous/Ripple counter were implemented and verify using truth table.

EXP NO: 11 DESIGN AND IMPLEMENTATION SYNCHRONOUS UP AND DOWN COUNTER USING IC 7476

AIM:

To design and Implement of synchronous counters and to verify its truth table.

APPARATUS REQUIRED:

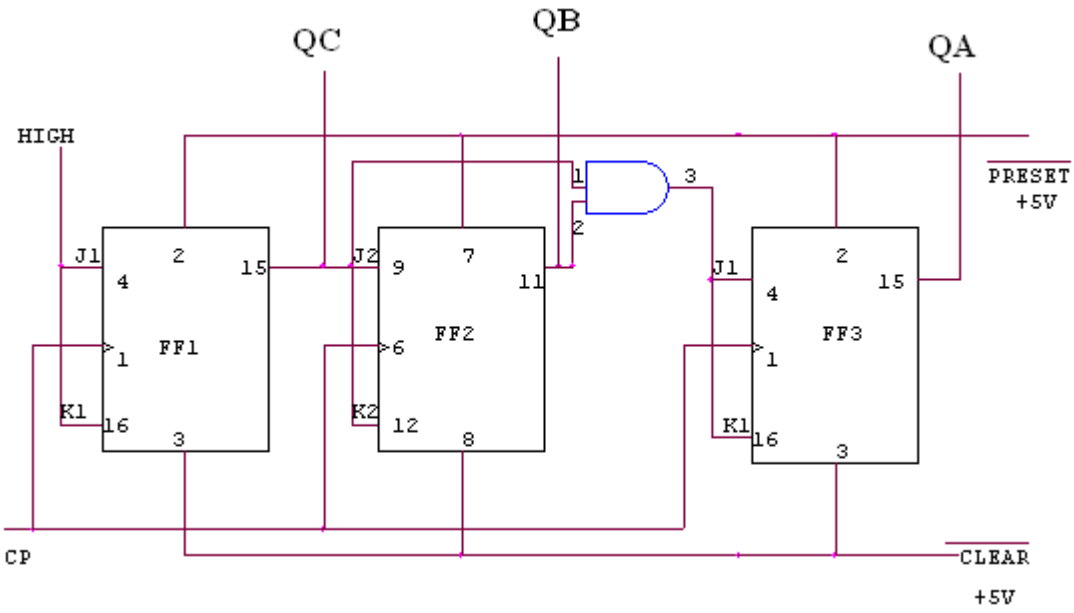
Sl. No.	COMPONENTS	SPECIFICATION/ RANGE	Qty
1.	JK FLIP FLOP	IC 7476	2
2.	AND GATE	IC 7408	1
3.	DIGITAL TRAINER KIT	-	1
4.	CONNECTING WIRES	-	Req.

3 BIT SYNCHRONOUS COUNTER

THEORY:

When a Counter is Clocked such that each Flip flop in the Counter is Triggered at the same time ,the counter is called Synchronous counter QA changes on each clock pulse as we progress from its original state to its final stack and then back to its original state .In Synchronous binary counter, the flip-flop in the lowest –order position is complemented with every pulse A flip-flop in any other position is complemented with a pulse provided all the bits in the lower order positions are equal to 1 . the binary count indicates that next higher order bit be complemented .Synchronous binary counters have a regular pattern and can be easily constructed with complementing flip-flops and gates .The counter could also be triggered on the positive edge of the pulse

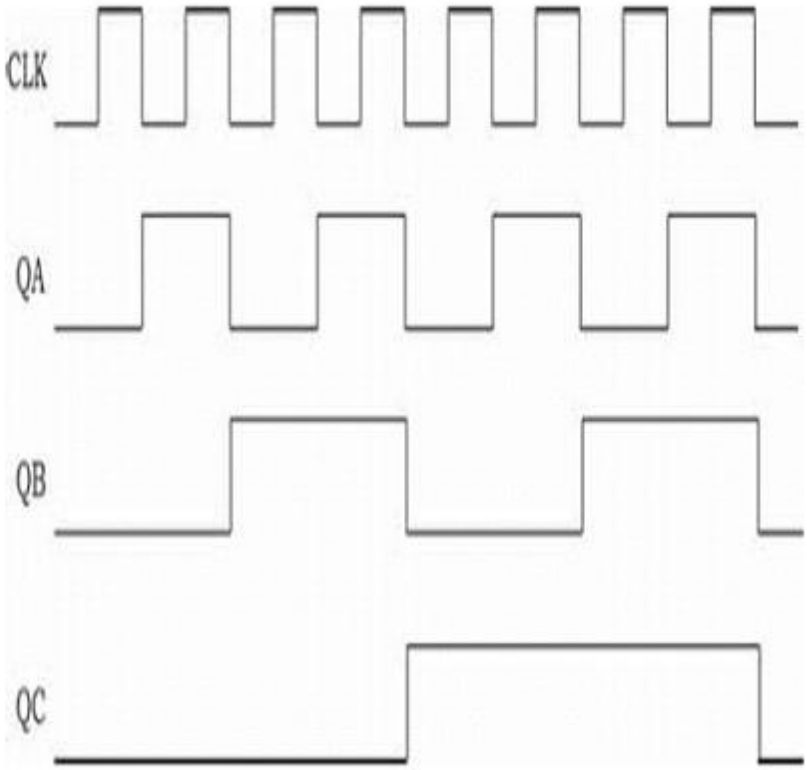
LOGIC DIAGRAM: 3-BIT SYNCHRONOUS UP COUNTER:



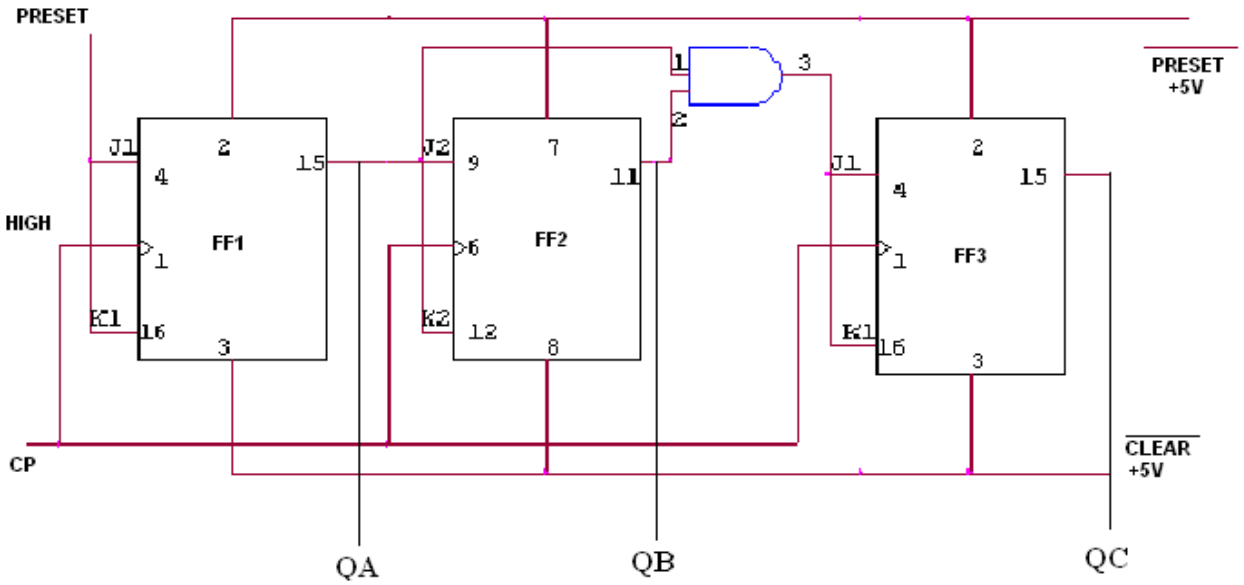
TRUTH TABLE:

3-bit synchronous up counter			
Clock	QC	QB	QA
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

TIMING DIAGRAM:

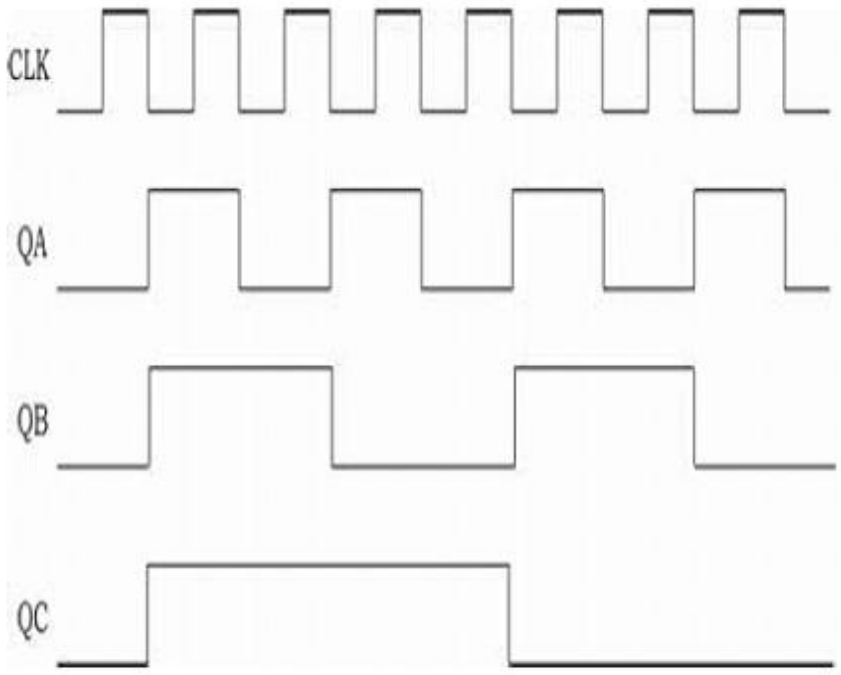


LOGIC DIAGRAM: 3-BIT SYNCHRONOUS DOWN COUNTER:



TRUTH TABLE: TIMING DIAGRAM:

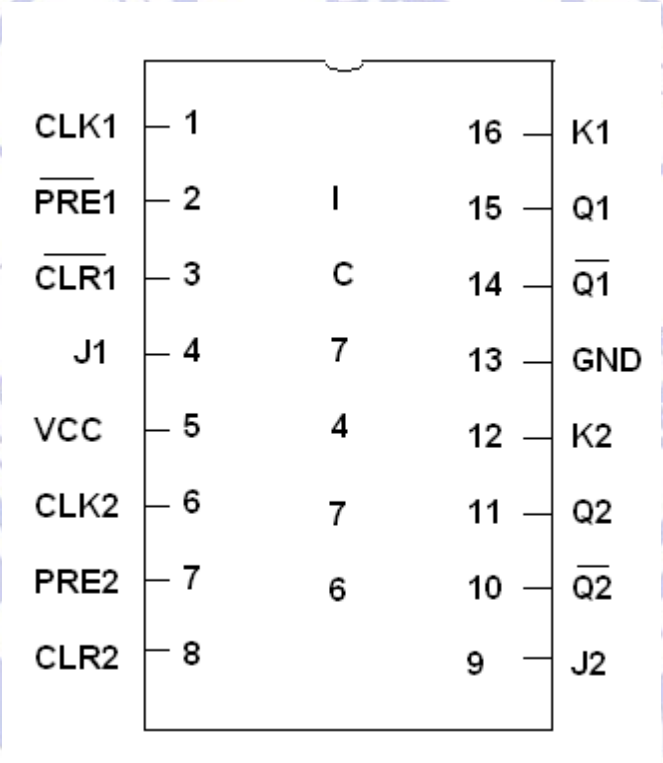
3-bit synchronous down counter			
Clock	QC	QB	QA
0	1	1	1
1	1	1	0
2	1	0	1
3	1	0	0
4	0	1	1
5	0	1	0
6	0	0	1
7	0	0	0
8	1	1	1
9	1	1	0



PROCEDURE:

1. The components are connected as shown in the circuit diagram.
2. The counter is set initially in the reset position.
3. Apply the clock pulse. It starts counting.
4. Continue giving the clock pulse and check for the output as given in the truth table.
5. The output is indicated by the LED display. The glowing of LED's confirming to required data in the truth table is verified and the conclusion is drawn.

PIN DIAGRAM FOR IC 7476:



Result:

Thus the 3-bit synchronous UP and DOWN counter were implemented using IC 7476 and verify using truth table.

EXP NO: 12 SIMULATION OF COMBINATIONAL CIRCUITS USING HDL

AIM:

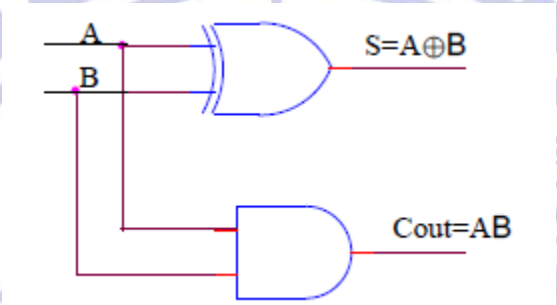
To write a verilog code for half adder, full adder and multiplexer.

TOOLS REQUIRED:

Xilinx 9.2

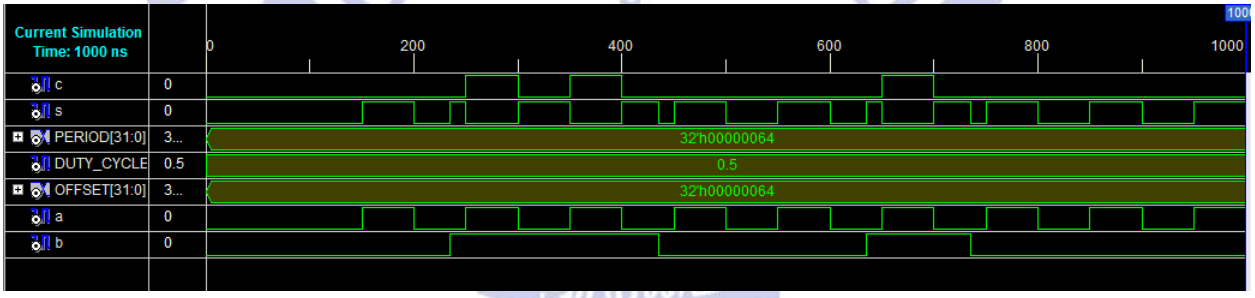
PROGRAM:

HALF ADDER

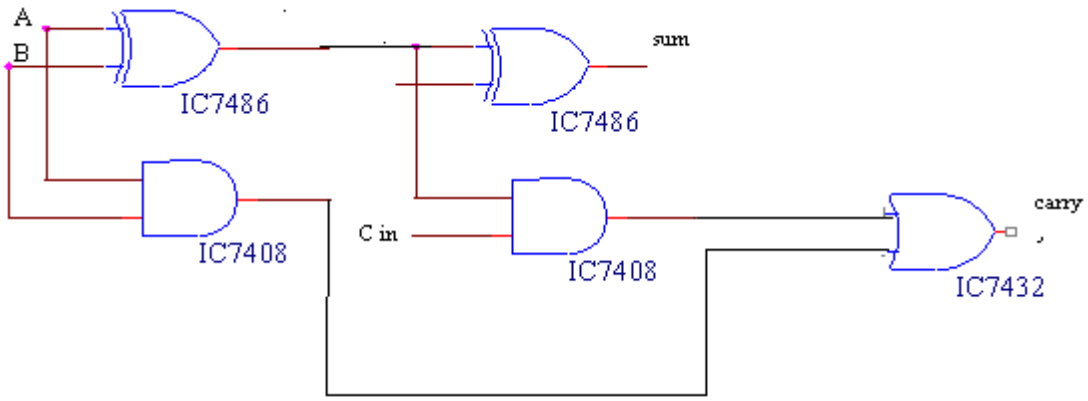



```
module halfadd(a, b, s, c);
    input a;
    input b;
    output s;
    output c;
    xor (s,a,b);
    and (c,a,b);
endmodule
```

SIMULATED OUTPUT:

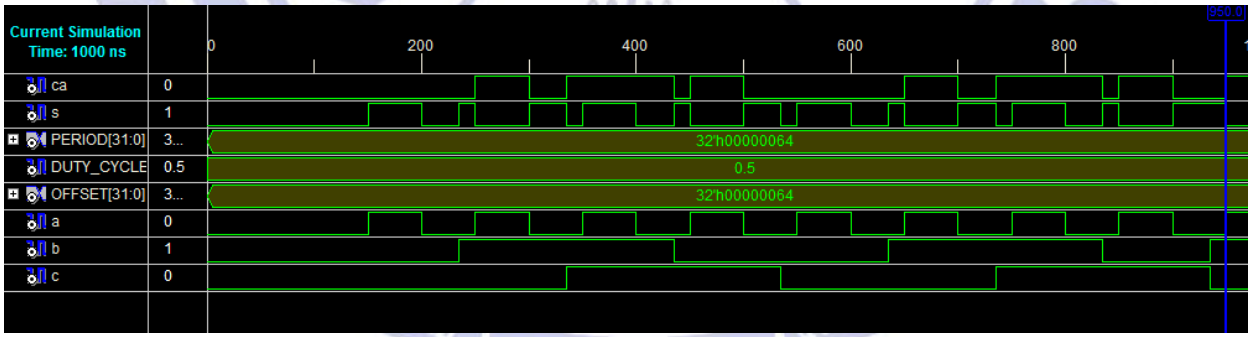


FULL ADDER:

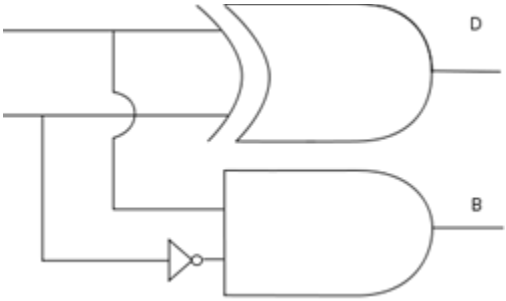



```
module fulladd(a, b, c, s, ca);
    input a;
    input b;
    input c;
    output s;
    output ca;
    wire  s1,c1,c2;
    xor g1(s1,a,b);
    and g2(c1,a,b);
    xor g3 (s,s1,c);
    and g4(c2,s1,c);
    or g5(ca,c1,c2);
endmodule
```

SIMULATED OUTPUT:

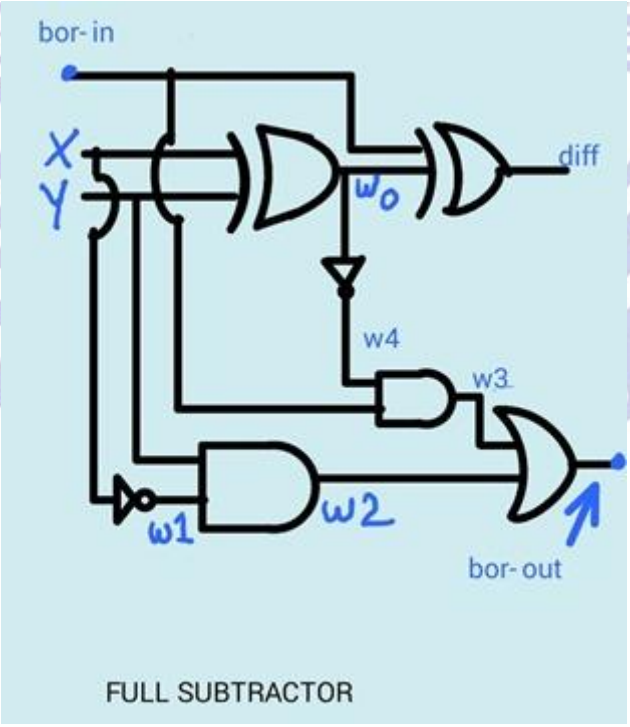


HALF SUBRACTOR:



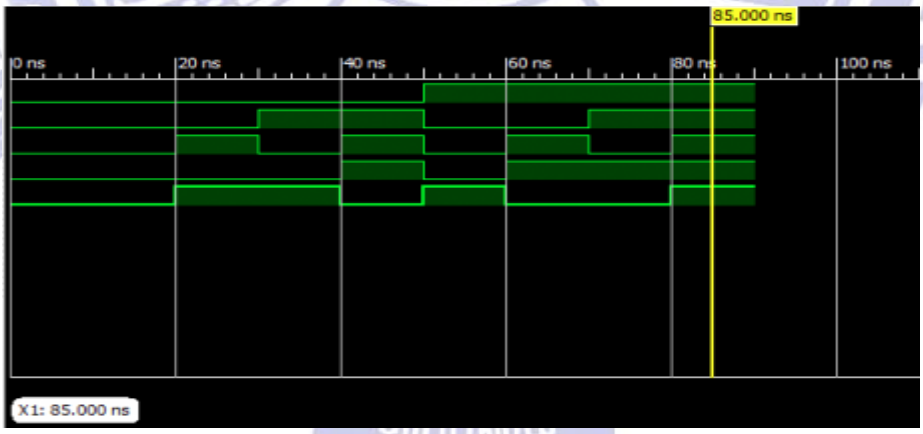

```
module half_subtractor(diff, bor, a, b);
// Port details
output diff, bor;
input a, b;
wire w0;
 //Gate implementation
xor xor1(diff,a,b);
and and1(bor,w0,b);
not not1(w0,a);
endmodule
```

FULL SUBTRACTOR:

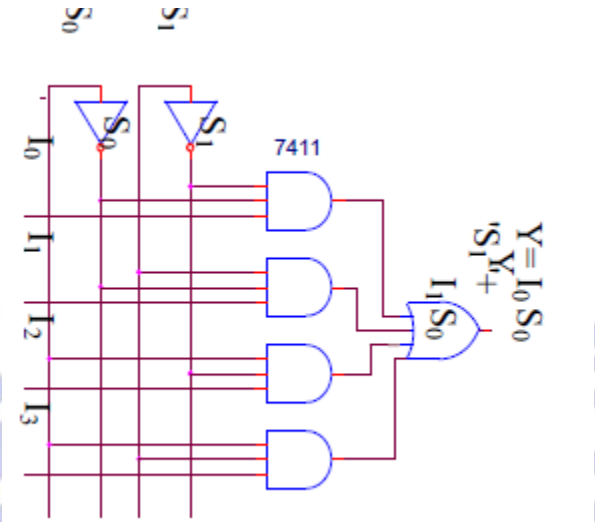


```
module full_subtractor_gate_level( diff, bor_out, x, y, bor_in);
 output bor_out,diff;
 input x,y,bor_in;
 wire w0,w1,w2,w3,w4;

 xor xor1 ( w0, x, y);
 xor xor2 ( diff, bor_in, w0);

 and and1 ( w2, y, w1);
 and and2 ( w3, bor_in, w4);

 not not1( w1, x);
 not not2( w4, w0);

 or or1( bor_out, w2, w3);

endmodule
```

SIMULATED OUTPUT:

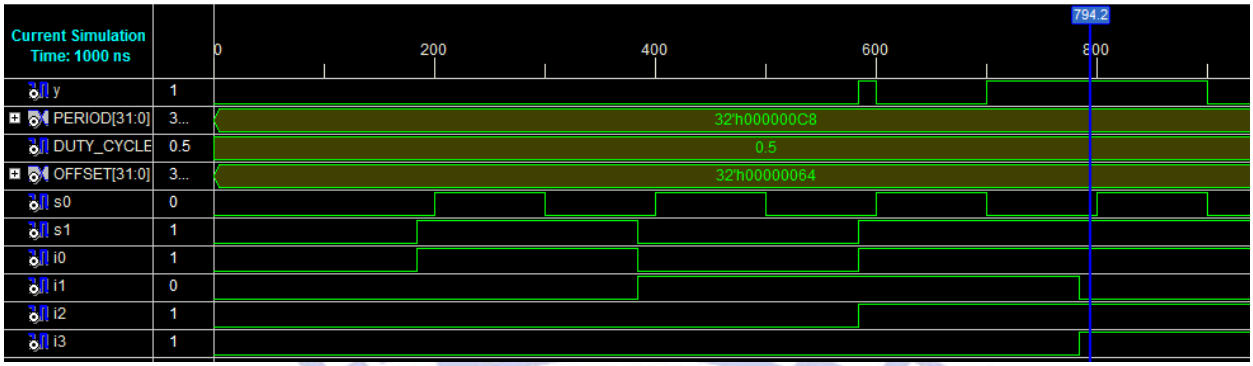


MULTIPLEXER:




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

SIMULATED OUTPUT:



RESULT:

Thus the verilog code for half adder, full adder and multiplexer were simulated and verified.

EXP NO: 13 SIMULATION OF SEQUENTIAL CIRCUITS USING HDL

AIM:

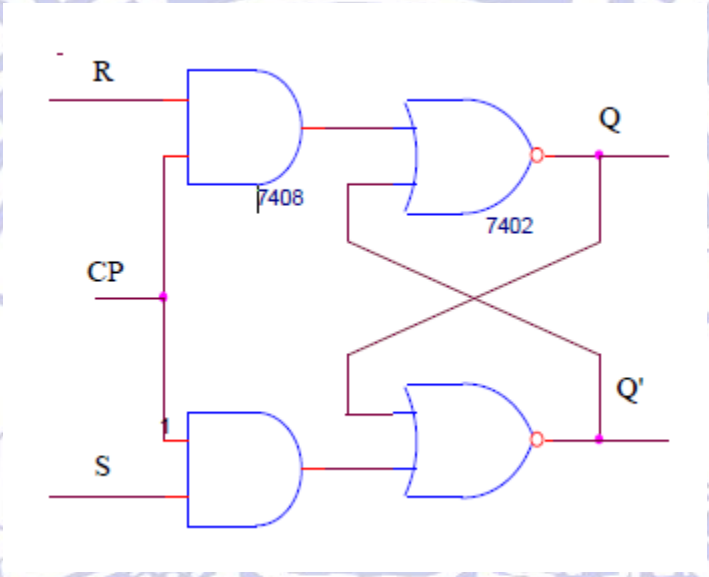
To write a verilog code for RS, D, JK flip flop and up counter

TOOLS REQUIRED:

Xilinx 9.2

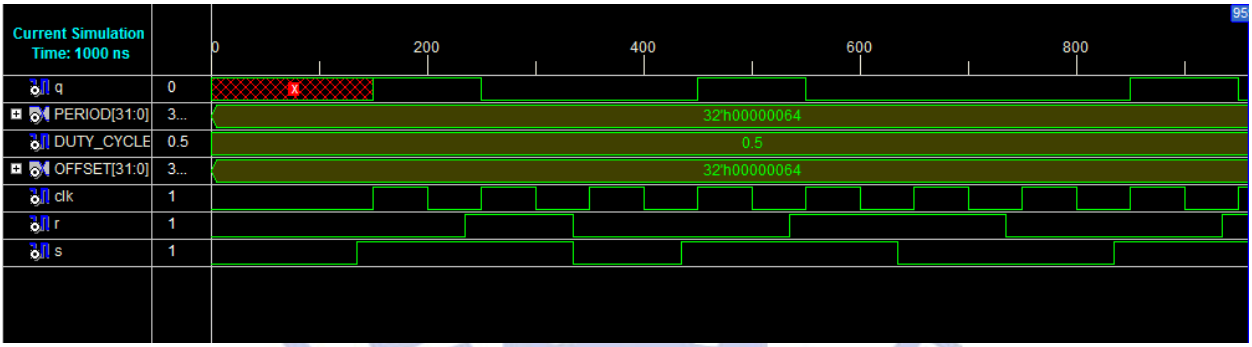
PROGRAM:

RS FLIP FLOP:

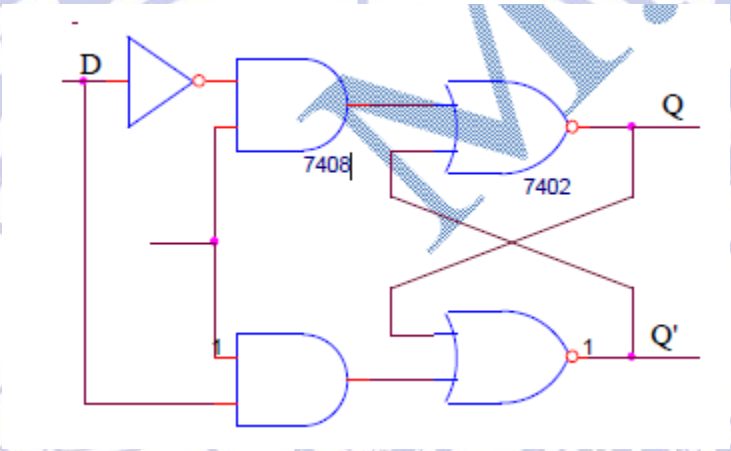



```
module sr1(clk, s, r, q);
    input clk;
    input s;
    input r;
    output reg q;
    always@(posedge clk or s or r)
    begin
    if(clk==1)
    if(s==1 && r==0)
    q<=1;
    else if(s==0 && r==0)
    q<=q;
    else
    q<=0;
    end
endmodule
```

SIMULATED OUTPUT:

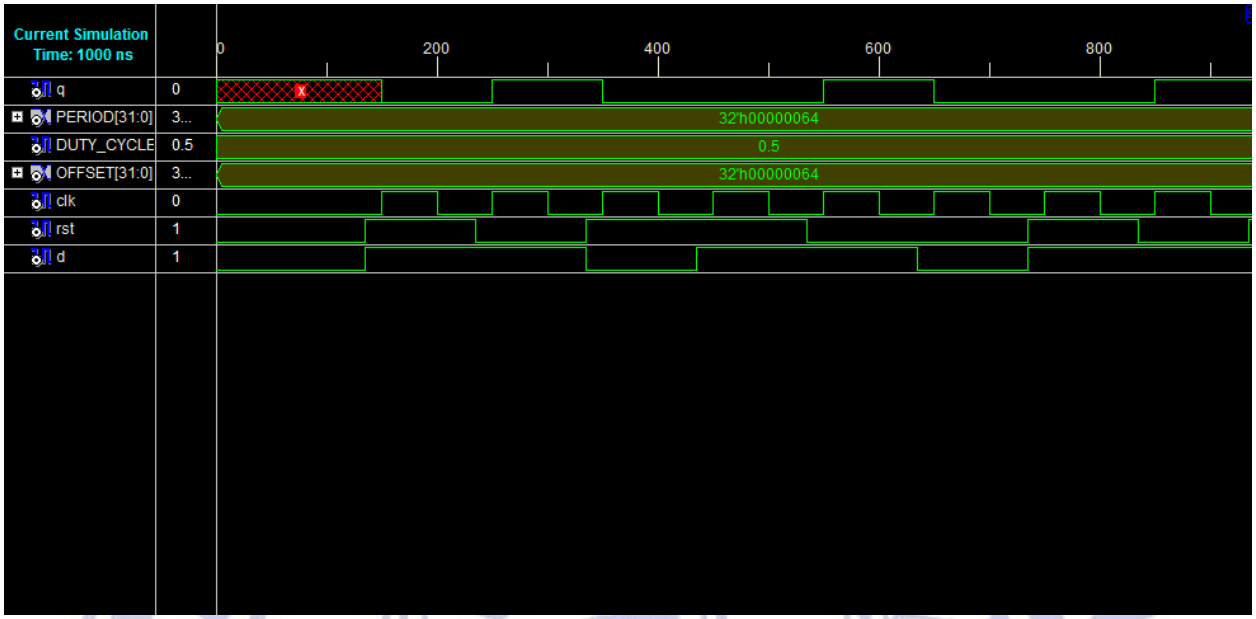


D FLIP FLOP:

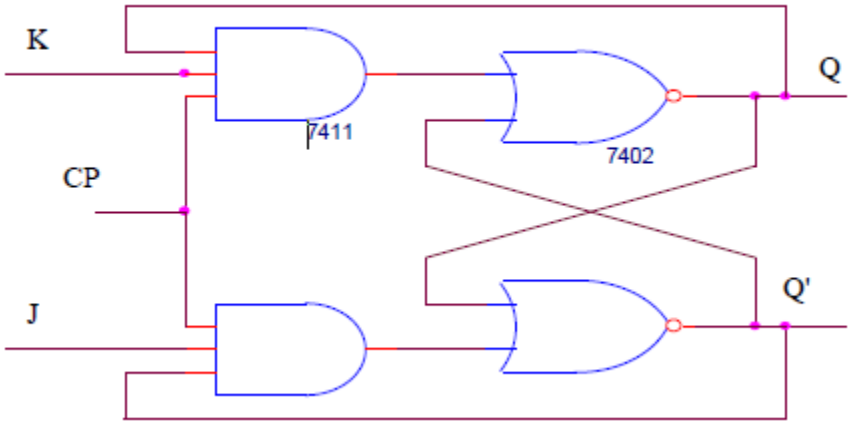



```
module dff1(d, clk, rst, q);
    input d;
    input clk;
    input rst;
    output q;
    reg q;
    always@(posedge clk)
    if(rst)
    q<=0;
    else
    q<=d;
endmodule
```

SIMULATED OUTPUT:

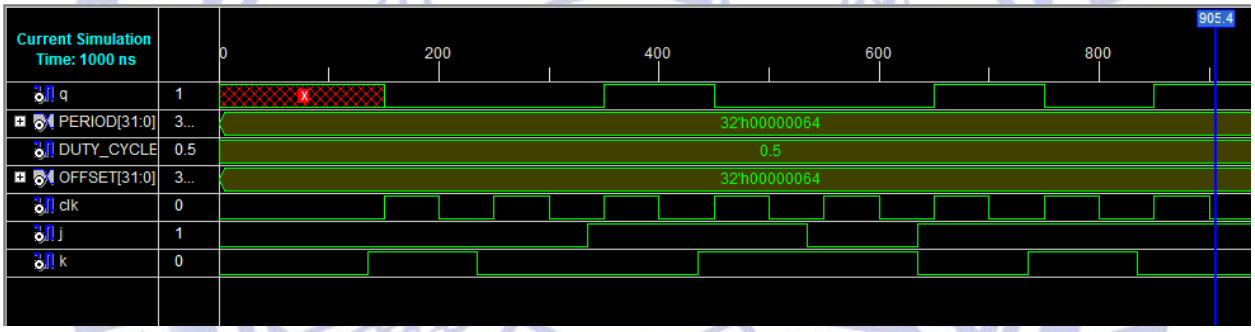


JK FLIP FLOP:



```
module jkflip(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 if(j==0 && k==1)
    q<=0;
    else
    q<=~q;
    end
endmodule
```

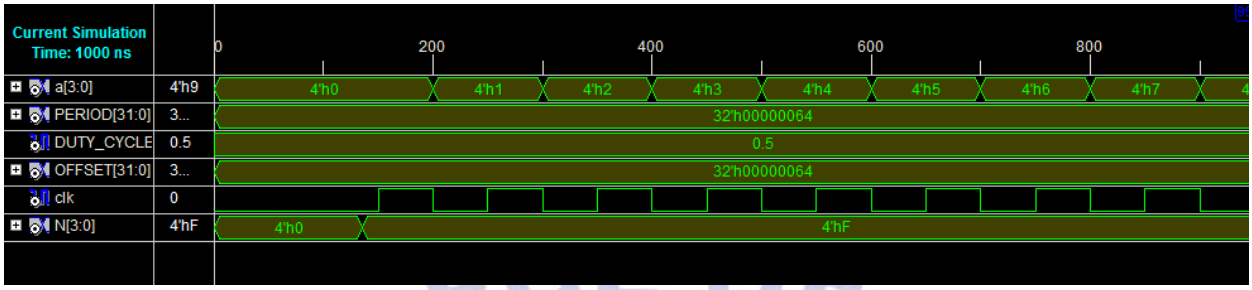
SIMULATED OUTPUT:



UP COUNTER:

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

SIMULATED OUTPUT:



RESULT:

Thus the verilog code for RS, D, JK flip flop and up counter were simulated and verified.

EX NO: 14 IMPLEMENTATION OF BOOLEAN FUNCTIONS

AIM:

To design the logic circuit and verify the truth table of the given Boolean expression,

$$F (A, B, C, D) = \Sigma (0, 1, 2, 5, 8, 9, 10)$$

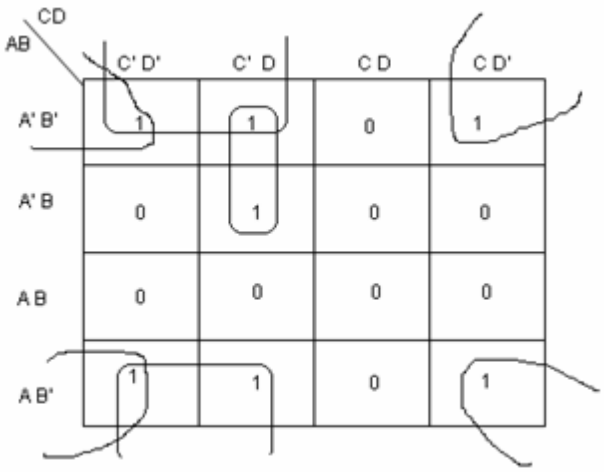
[Design can be changed by changing the Boolean expression]

APPARATUS REQUIRED:

S.NO	COMPONENT	SPECIFICATION	QTY.
1.	IC TRAINER KIT		1
2.	AND GATE	IC 7408	3
3.	OR GATE	IC 7432	2
4.	NOT GATE	IC 7404	4
5.	EX-OR GATE	IC 7486	
6.	PATCH CORDS	-	REQUIRED

DESIGN:

Given , F (A,B,C,D) = Σ (0,1,2,5,8,9,10)
The output function F has four input variables hence a four variable Karnaugh Map is used to obtain a simplified expression for the output as shown,



From the K-Map,

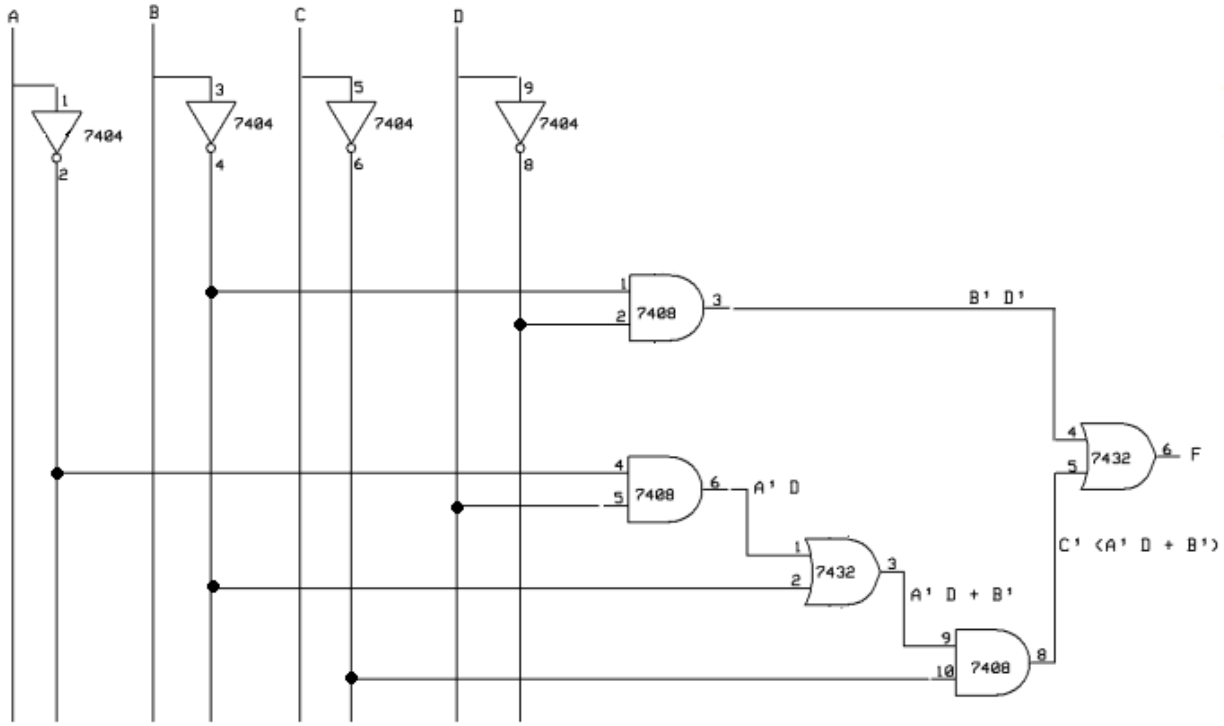
$$F = B' C' + D' B' + A' C' D$$

Since we are using only two input logic gates the above expression can be re-written as,

$$F = C' (B' + A' D) + D' B'$$

Now the logic circuit for the above equation can be drawn.

CIRCUIT DIAGRAM:



TRUTH TABLE:

S.No	INPUT				OUTPUT
	A	B	C	D	F=D'B'+C'(B'+A'D)
1.	0	0	0	0	1
2.	0	0	0	1	1
3.	0	0	1	0	1
4.	0	0	1	1	0
5.	0	1	0	0	0
6.	0	1	0	1	1
7.	0	1	1	0	0
8.	0	1	1	1	0
9.	1	0	0	0	1
10.	1	0	0	1	1

11.	**1**	**0**	**1**	**0**	**1**
12.	**1**	**0**	**1**	**1**	**0**
13.	**1**	**1**	**0**	**0**	**0**
14.	**1**	**1**	**0**	**1**	**0**
15.	**1**	**1**	**1**	**0**	**0**
16.	**1**	**1**	**1**	**1**	**0**

PROCEDURE:

1. Connections are given as per the circuit diagram
2. For all the ICs 7th pin is grounded and 14th pin is given +5 V supply.
3. Apply the inputs and verify the truth table for the given Boolean expression.

RESULT:

The truth table of the given Boolean expression was verified.