SIMATS ENGINEERING

Engineer to Excel

SIMATS

Saveetha Institute of Medical And Technical Sciences

(Declared as Deemed to be University under Section 3 of UGC Act 1956)

Department of Electronics and Communication Engineering

ECA0206 DIGITAL CIRCUITS LAB RECORD

Name:	
Register Number:	

INDEX

Ex No	Date	Title of the Experiment	Page No	Marks	Sign
1.		Study of Logic gates			
2.		Verification of Boolean theorems using logic gates			
3.		Implementation of a combinational circuit for an arbitrary function			
4.		Design and implementation of code converters using logic gate			
5.		Design and implementation of half adder and full adder using logic gates			
6.		Design and implementation of half subtractor and full subtractor using logic gates			
7.		Design and implementation of multiplexers			
8.		Design and implementation of decoder			
9.		Design and implementation of flip-flops			
10.		Design of 2–bit synchronous counter			
11.		Design and implementation of shift registers			
12.		Simulation of logic gates using VHDL			
13.		Simulation of adder circuits using VHDL			
14.		Simulation of subtractor circuits using VHDL			
15.		Simulation of multiplexer and demultiplexer using VHDL			
16.		Simulation of encoder and decoder using VHDL			

STUDY OF LOGIC GATES

DATE:

AIM:

To study logic gates and verify their truth tables of all the logic functions.

APPARATUS REQUIRED:

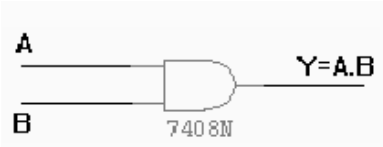
SL No.	COMPONENT	SPECIFICATION	QTY
1.	AND GATE	IC 7408	1
2.	OR GATE	IC 7432	1
3.	NOT GATE	IC 7404	1
4.	NAND GATE 2 I/P	IC 7400	1
5.	NOR GATE	IC 7402	1
6.	X-OR GATE	IC 7486	1
8.	IC TRAINER KIT	-	1
9.	CONNECTING WIRES		

PROCEDURE:

(i) Connections are given as per the circuit diagram.
(ii) Logical inputs are given as per the circuit diagram.
(iii) Observe the output and verify the truth table.

AND GATE:

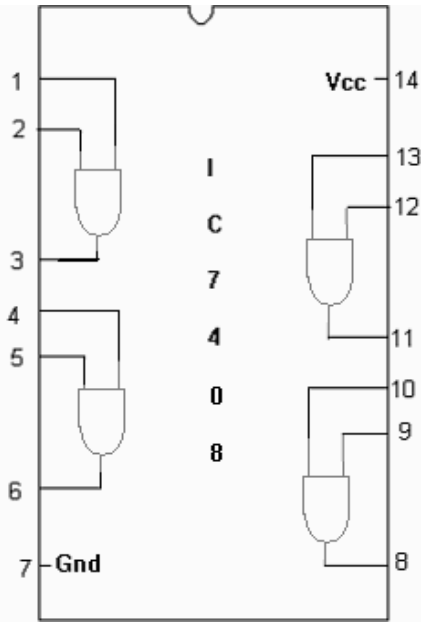
SYMBOL:



TRUTH TABLE

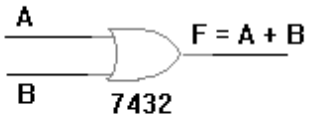
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

PIN DIAGRAM:

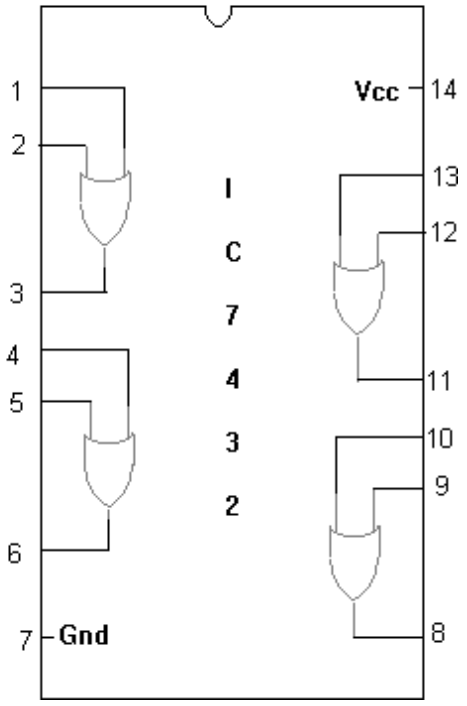


OR GATE:

SYMBOL :



PIN DIAGRAM :

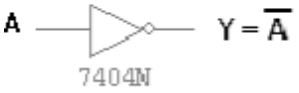


TRUTH TABLE

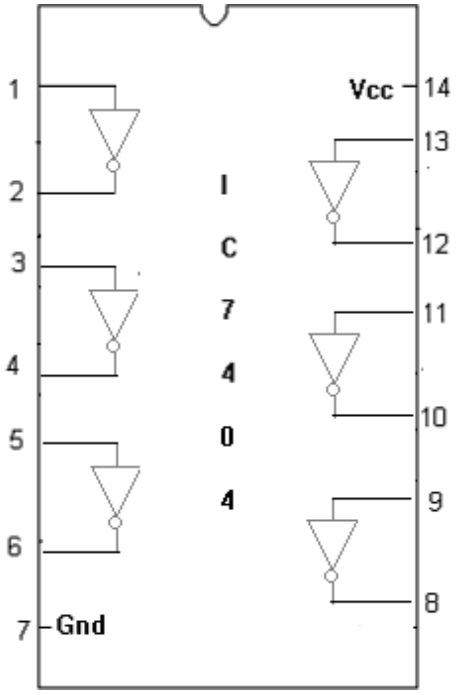
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

NOT GATE:

SYMBOL:



PIN DIAGRAM:

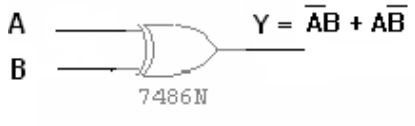


TRUTH TABLE :

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

X-OR GATE:

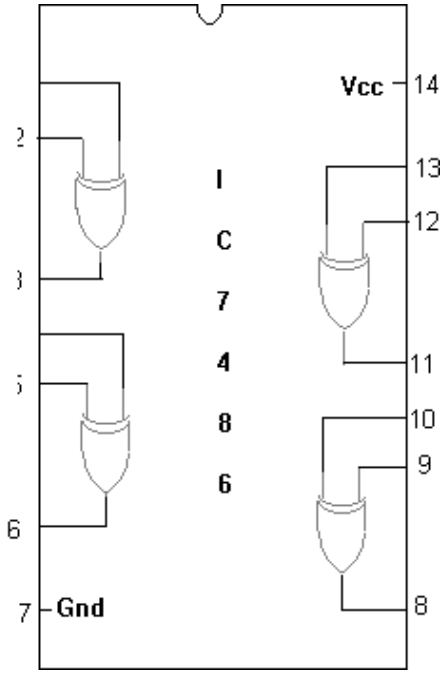
SYMBOL:



TRUTH TABLE :

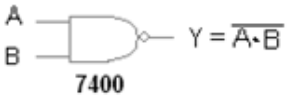
A	B	$\overline{A}B + A\overline{B}$
0	0	0
0	1	1
1	0	1
1	1	0

PIN DIAGRAM :



2-INPUT NAND GATE:

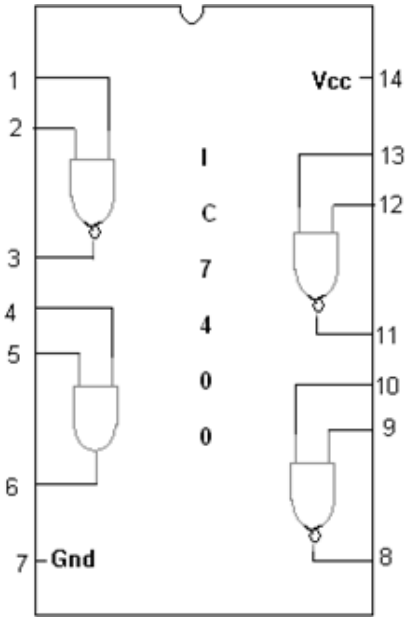
SYMBOL:



TRUTH TABLE

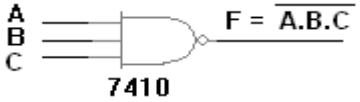
A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

PIN DIAGRAM:

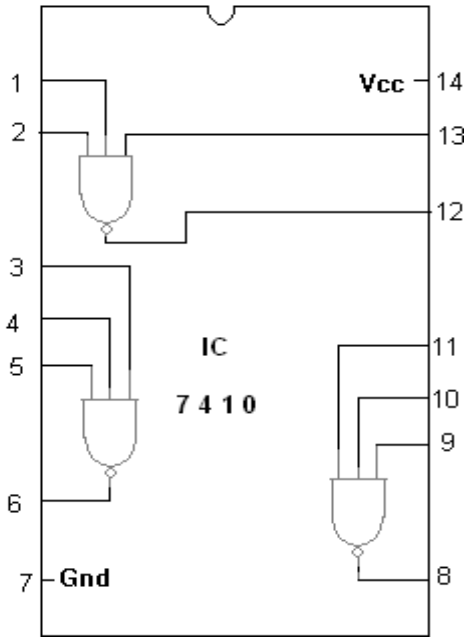


3-INPUT NAND GATE :

SYMBOL :



PIN DIAGRAM :

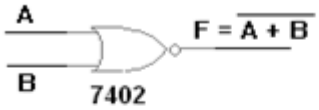


TRUTH TABLE

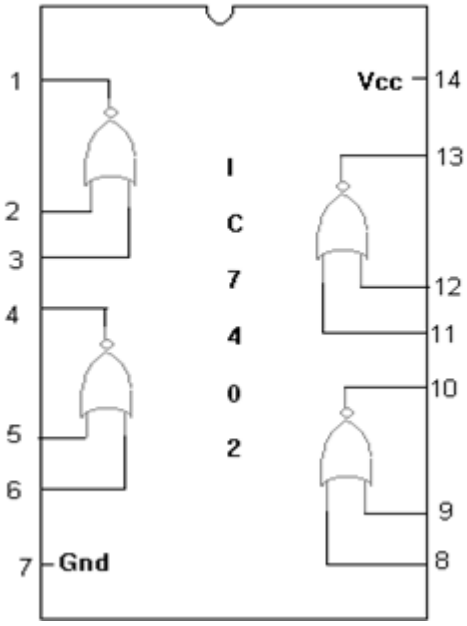
A	B	C	$\overline{A.B.C}$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

NOR GATE:

SYMBOL :



PIN DIAGRAM :



TRUTH TABLE

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

RESULT:

Thus, the gates (AND, OR, NOT, NAND, NOR, XOR, and XNOR GATES) are studied and verified using the Truth table.

V VERIFICATION OF BOOLEAN THEOREMS USING LOGIC GATES

DATE:

AIM:

To verify Boolean theorems using logic gates

APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	AND GATE	IC 7408	1
2.	NOT GATE	IC 7404	1
3.	OR GATE	IC 7432	1
4.	IC TRAINER KIT	-	1
5.	CONNECTING WIRES		

PROCEDURE:

a. Connections are given as per the logic diagram
b. Input is given to the circuit making high '1' i.e., +5 or Vcc Supply to the 14th pin, and for low '0' i.e., GND to the 7th pin of Gate IC
c. Depending upon the truth table, if the LED Glow represents 1 and else it represents '0'
d. Verify the truth table as given
e. Repeat the procedure steps for different theorems.

BOOLEAN THEOREM:

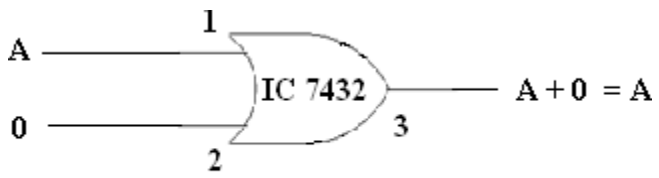
POSTULATES	I	II
Identity	A + 0 = A	A. 1 = A
Commutative	A + B = B + A	AB = BA
Distributive	A (B + C) = AB + AC	A + BC = (A + B) (A + C)
Complement	A + A' =1	A.A' = 0
Idempotency	A + A = A	A.A = A
	A + 1 = 1	A.0 = 0
Involution	(A')' = A	
Absorption	A + AB = A	A (A + B) = A
	A + A'B = A + B	A. (A'+ B) = AB
Associative	A + (B + C) = (A + B) + C	A (B C) = (AB) C
De Morgan's Law	(A + B)' =A'. B'	(AB)' = A'+ B'

1. **IDENTITY**

TRUTH TABLE

I/P		O/P
A	**0**	**A+0= A**
0	0	**0**
1	0	**1**

LOGIC DIAGRAM

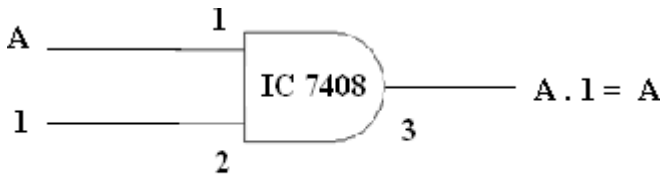


Identity law A + 0 = A

TRUTH TABLE

I/P		O/P
A	**0**	**A.1=A**
0	0	0
1	0	1

LOGIC DIAGRAM



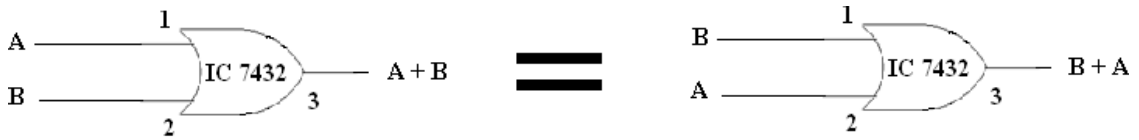
Identity law A . 1 = A

2. **COMMUTATIVE**

TRUTH TABLE

I/P		LHS (O/P)	RHS (O/P)
A	B	A+B	B+A
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

LOGIC DIAGRAM

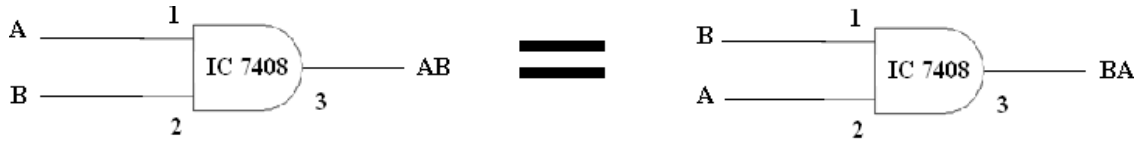


Commulative law A + B = B + A

TRUTH TABLE

I/P		LHS (O/P)	RHS (O/P)
A	B	A. B	B. A
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

LOGIC DIAGRAM



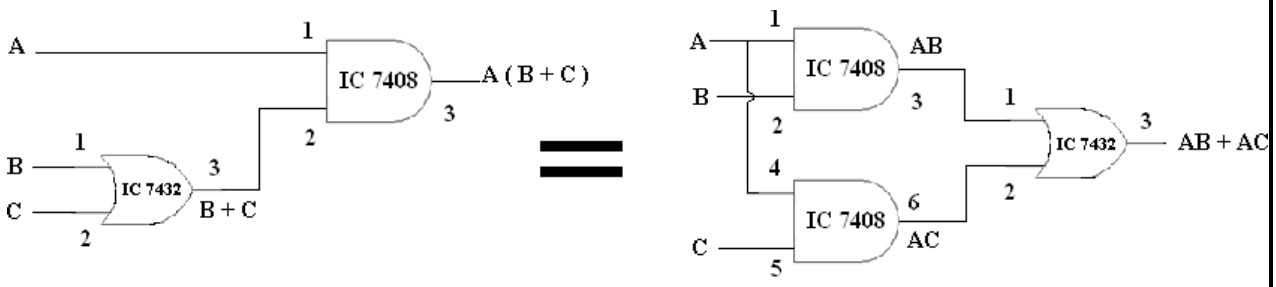
Commulative law AB = BA

3. DISTRIBUTIVE LAW

TRUTH TABLE

I/P				LHS (O/P)			RHS (O/P)
A	**B**	**C**	**B+C**	**A(B+C)**	**AB**	**AC**	**AB+AC**
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

LOGIC DIAGRAM

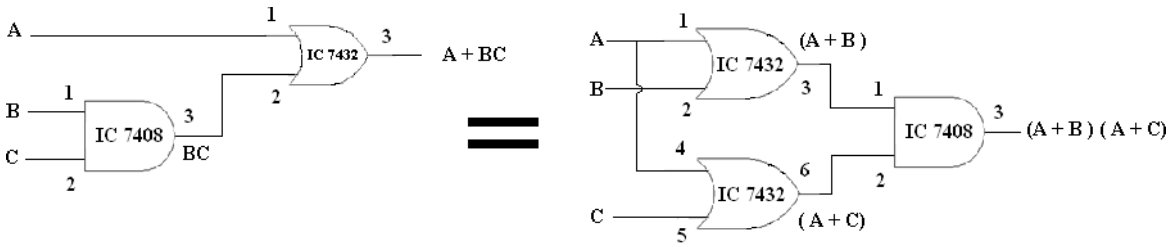


Distributive 1 A (B + C) = AB + AC

TRUTH TABLE

I/P				LHS (O/P)			RHS (O/P)
A	**B**	**C**	**BC**	**A+BC**	**A+B**	**A+C**	**(A+B) (A+C)**
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

LOGIC DIAGRAM



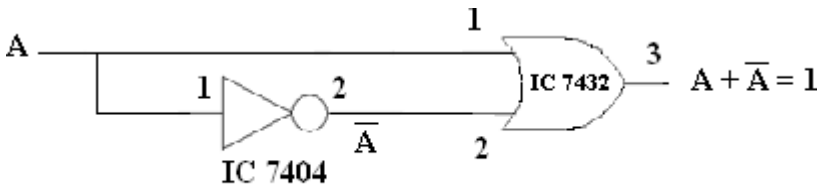
Distributive 2 A + BC = (A + B) (A + C)

4. **<u>COMPLEMENT</u>**

TRUTH TABLE

I/P		O/P
A	A'	A+A'
0	1	1
1	0	1

LOGIC DIAGRAM

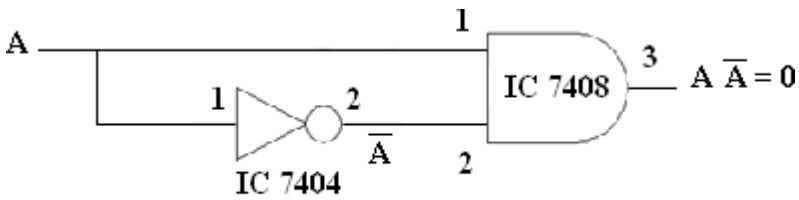


Complement 1 A + A' =1

TRUTH TABLE

I/P		O/P
A	A'	AA'
0	1	0
1	0	0

LOGIC DIAGRAM



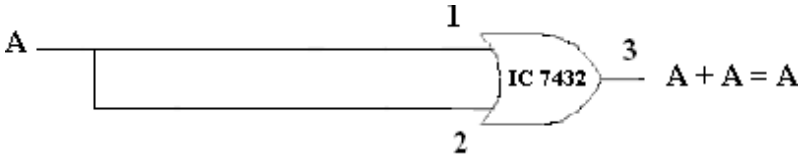
Complement 2 A.A' = 0

5. **<u>IDEMPOTENCY</u>**

TRUTH TABLE

I/P		O/P
A	A	A+A
0	0	0
1	1	1

LOGIC DIAGRAM

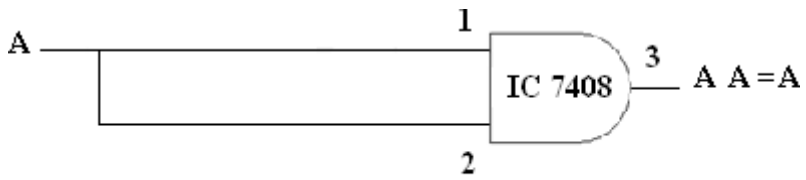


Idempotency 1 A + A = A

TRUTH TABLE

I/P		O/P
A	A	AA
0	0	0
1	1	1

LOGIC DIAGRAM

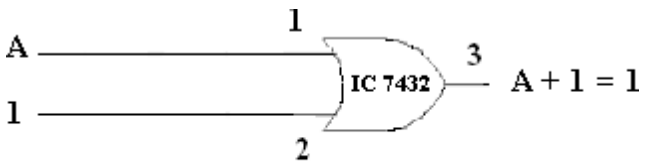


Idempotency 2 A.A = A

TRUTH TABLE

I/P		O/P
A	**1**	**A+1**
0	1	1
1	1	1

LOGIC DIAGRAM

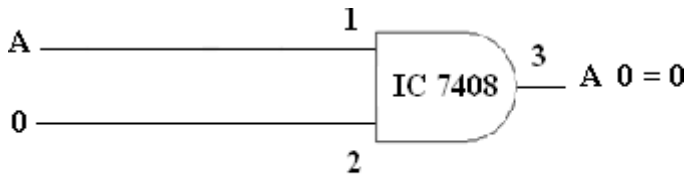


Idempotency 3 A + 1 = 1

TRUTH TABLE

I/P		O/P
A	**0**	**A.0**
0	0	0
1	0	0

LOGIC DIAGRAM



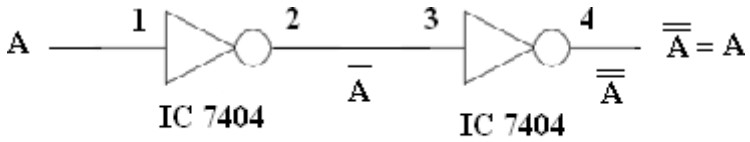
Idempotency 4 A.0 = 0

6. INVOLUTION

TRUTH TABLE

I/P		O/P
A	**A'**	**(A')'**
0	1	**0**
1	0	**1**

LOGIC DIAGRAM



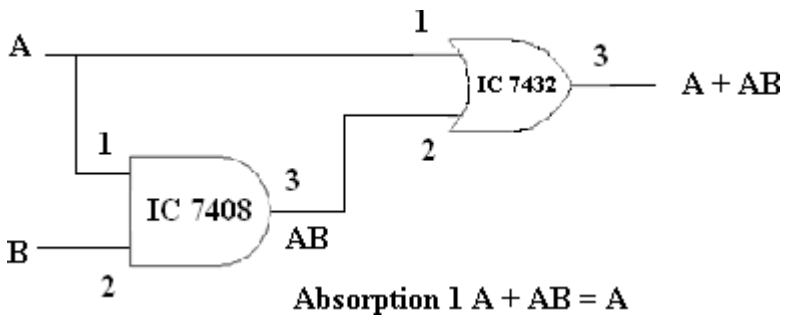
Involution (A')' = A

7. ABSORPTION

TRUTH TABLE

I/P			O/P
A	**B**	**AB**	**A + AB**
0	0	0	**0**
0	1	0	**0**
1	0	0	**1**
1	1	1	**1**

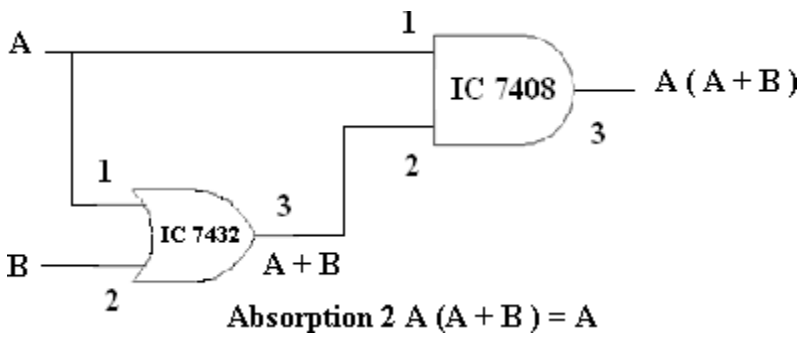
LOGIC DIAGRAM



TRUTH TABLE

I/P			O/P
A	B	A+B	**A(A + B)**
0	0	0	**0**
0	1	1	**0**
1	0	1	**1**
1	1	1	**1**

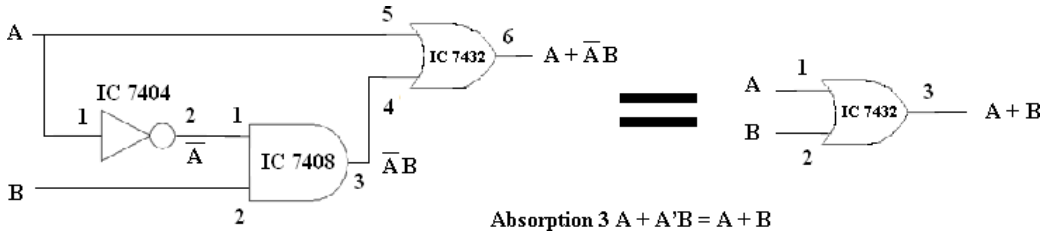
LOGIC DIAGRAM



TRUTH TABLE

I/P				LHS (O/P)	RHS (O/P)
A	B	A'	A'B	A + (A'B)	A + B
0	0	1	0	0	0
0	1	1	1	1	1
1	0	0	0	1	1
1	1	0	0	1	1

LOGIC DIAGRAM

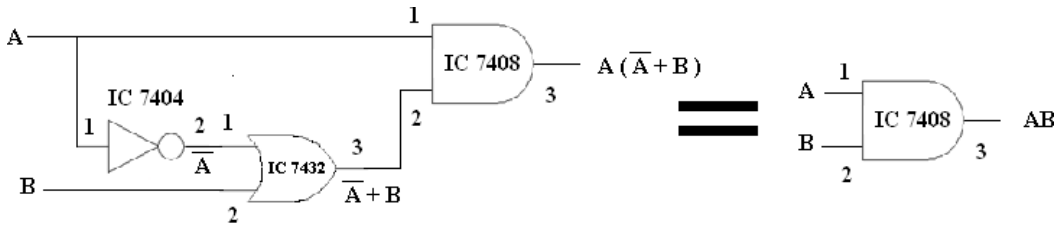


Absorption 3 A + A'B = A + B

TRUTH TABLE

I/P				LHS (O/P)	RHS (O/P)
A	B	A'	A' + B	A (A' + B)	AB
0	0	1	1	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	1	0	1	1	1

LOGIC DIAGRAM



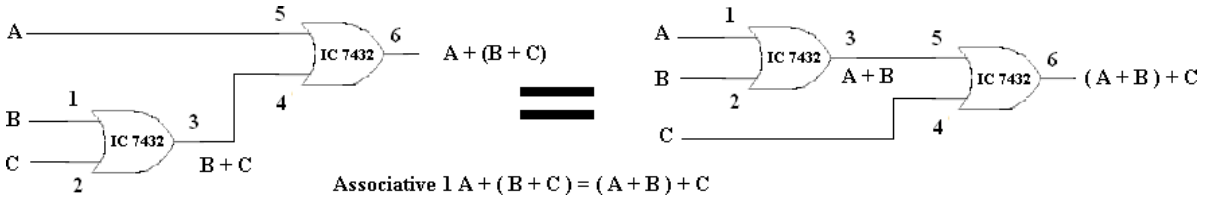
Absorption 4 A. (A'+ B) = AB

8. **<u>ASSOCIATIVE</u>**

TRUTH TABLE

I/P				LHS (O/P)		RHS (O/P)
A	**B**	**C**	**B+C**	**A + (B+C)**	**A + B**	**(A + B) + C**
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

LOGIC DIAGRAM

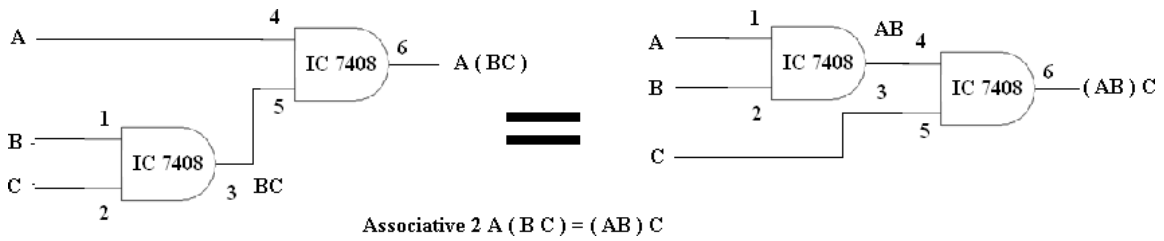


Associative 1 A + (B + C) = (A + B) + C

TRUTH TABLE

I/P				LHS (O/P)		RHS (O/P)
A	B	C	BC	A (BC)	AB	(AB) C
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	0	1	0
1	1	1	1	1	1	1

LOGIC DIAGRAM



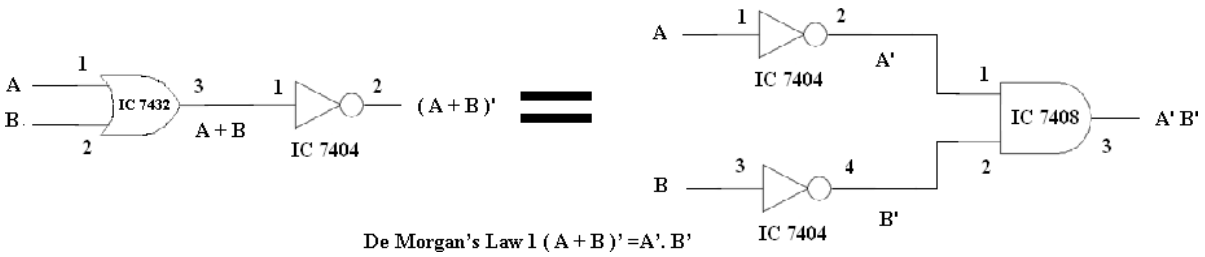
Associative 2 A (B C) = (AB) C

9. **DE MORGAN'S LAW**

TRUTH TABLE

I/P			LHS (O/P)			RHS (O/P)
A	B	(A + B)	(A + B)'	A'	B'	A'.B'
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

LOGIC DIAGRAM

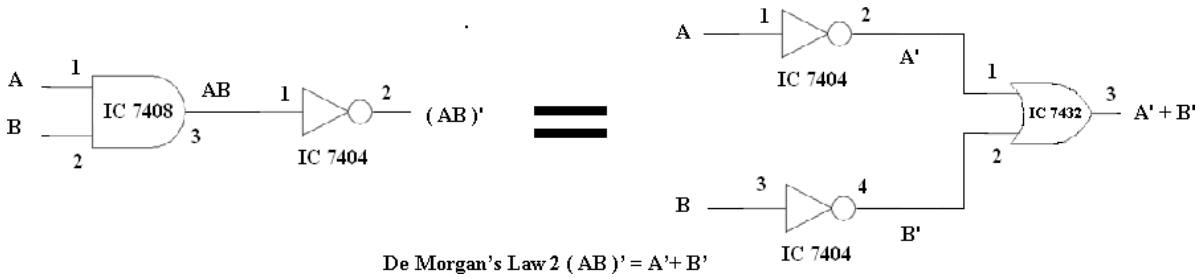


De Morgan's Law 1 (A + B)' =A'. B'

TRUTH TABLE

I/P			LHS (O/P)			RHS (O/P)
A	B	(AB)	(AB)'	A'	B'	A' + B'
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

LOGIC DIAGRAM



De Morgan's Law 2 (AB)' = A'+ B'

RESULT:

The different theorems of Boolean algebra and Demorgan's theorem are designed and verified using logic gates.

IMPLEMENTATION OF COMBINATION CIRCUIT FOR AN ARBITRARY FUNCTION

DATE:

AIM:

To Realize the Boolean Expression AB + BC + AC using Logic gates and verify its Performance with its truth table and the Digital Trainer kit. **Y= AC + AB + BC**

APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	AND GATE	IC 7408	1
2.	OR GATE	IC 7432	1
3.	IC TRAINER KIT	-	1
4.	CONNECTING WIRES	-	few

THEORY:

Combinational logic circuits are circuits in which the output at any time depends upon the combination of input signals present at that instant only, and does not depend on any past conditions. The combinational circuit block can be considered as a network of logic gates that accept signals from inputs and generate signals to outputs

PROCEDURE:

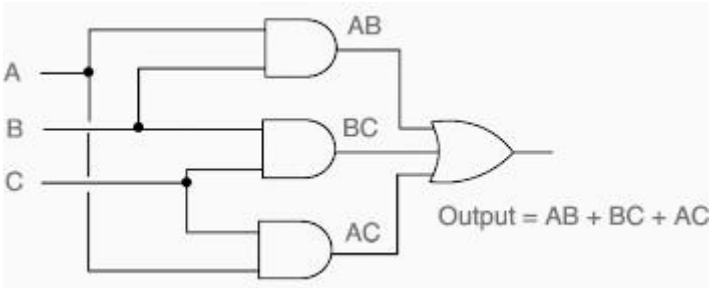
a. Connections are given as per the logic diagram

b. The input is given to the circuit, making high ‘1’, i.e., +5 or Vcc Supply to the 14th pin, and low ‘0’, i.e., GND to the 7th pin of the Gate IC.

c. c. Depending upon the truth table, if the LED Glows, it represents 1, and otherwise, it represents ‘0’.

d. Verify the truth table as given

LOGIC DIAGRAM:



TRUTH TABLE:

I/P						O/P
A	**B**	**C**	**AB**	**BC**	**AC**	**AB + BC + AC**
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	1
1	0	0	0	0	0	0
1	0	1	0	0	1	1
1	1	0	1	0	0	1
1	1	1	1	1	1	1

RESULT:

Thus, the given Arbitrary function AB+BC=AC was verified using truth table.

DESIGN AND IMPLEMENTATION OF CODE CONVERTERS USING LOGIC GATES

DATE:

AIM:

To design and implement a 4-bit code converter for the following

a. Binary to gray code converter
b. Gray to binary code converter

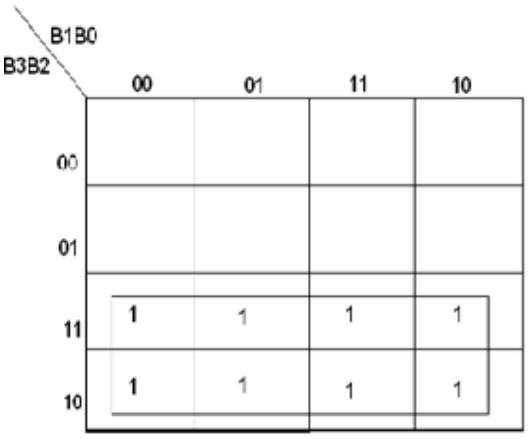
APPARATUS REQUIRED:

Sl. No.	COMPONENT	SPECIFICATION	QTY
1.	X-OR GATE	IC 7486	1
5.	IC TRAINER KIT	-	1
6.	CONNECTING WIRES	-	

DESIGN FOR BINARY TO GRAY CODE CONVERTERTRUTH TABLE:

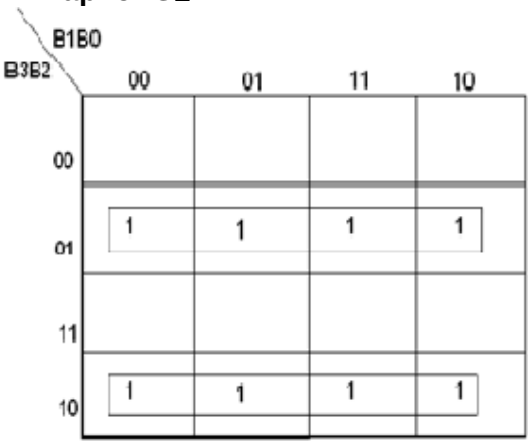
Binary input				**Gray code output**			
B3	**B2**	**B1**	**B0**	**G3**	**G2**	**G1**	**G0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**1**	**0**	**0**	**0**	**1**
0	**0**	**1**	**0**	**0**	**0**	**1**	**1**
0	**0**	**1**	**1**	**0**	**0**	**1**	**0**
0	**1**	**0**	**0**	**0**	**1**	**1**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



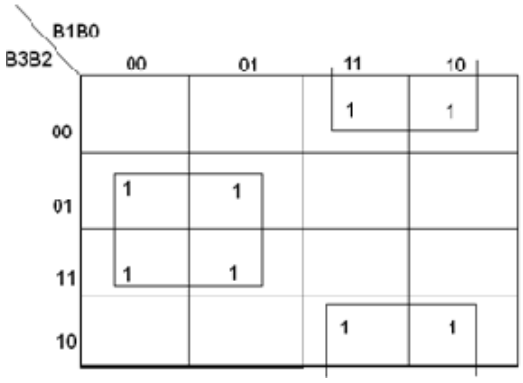
G3 = B3

K-Map for G2:



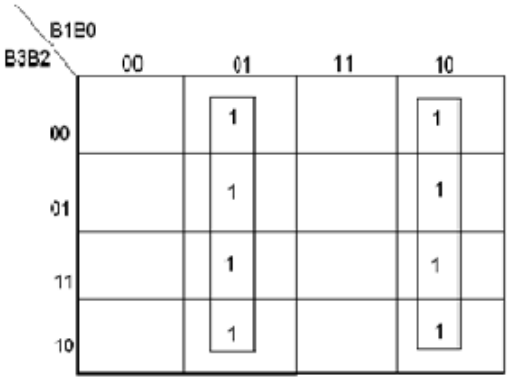
G2 = B3⊕B2

K-Map for G1:



G1 = B1⊕B2

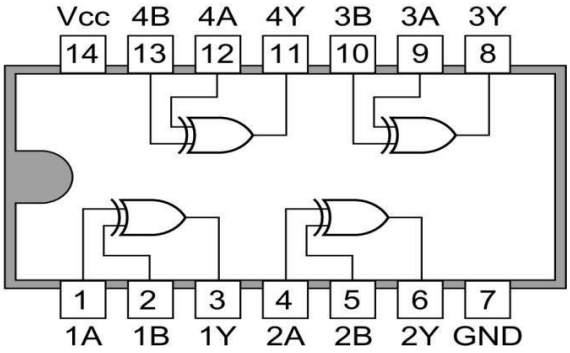
K-Map for G0:



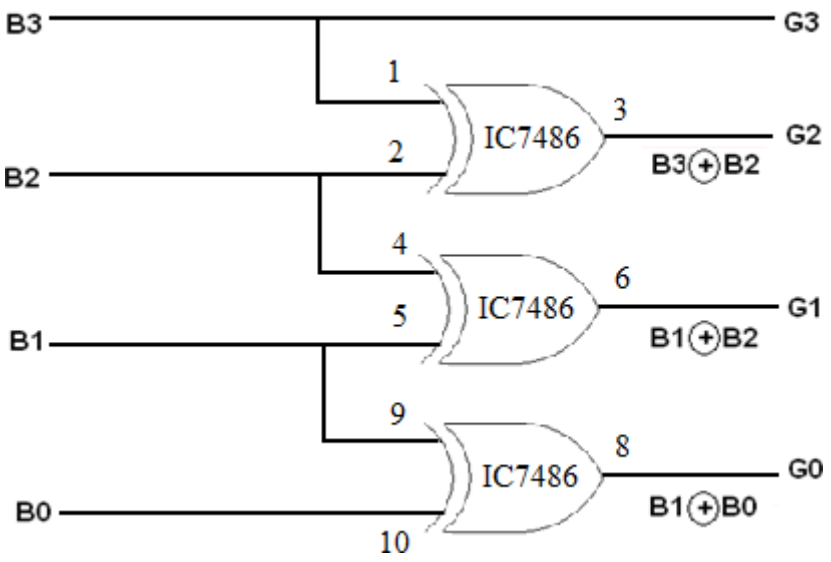
G0 = B1⊕B0

PIN DIAGRAM:

7486 Quad 2-input ExOR Gates



LOGIC DIAGRAM FOR BINARY TO GRAY CODE CONVERTOR:



DESIGN FOR GRAY TO BINARY CODE CONVERTER 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 B3

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

B3 = G3

K-Map for B2

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

B2 = G3⊕G2

K-Map for B1

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

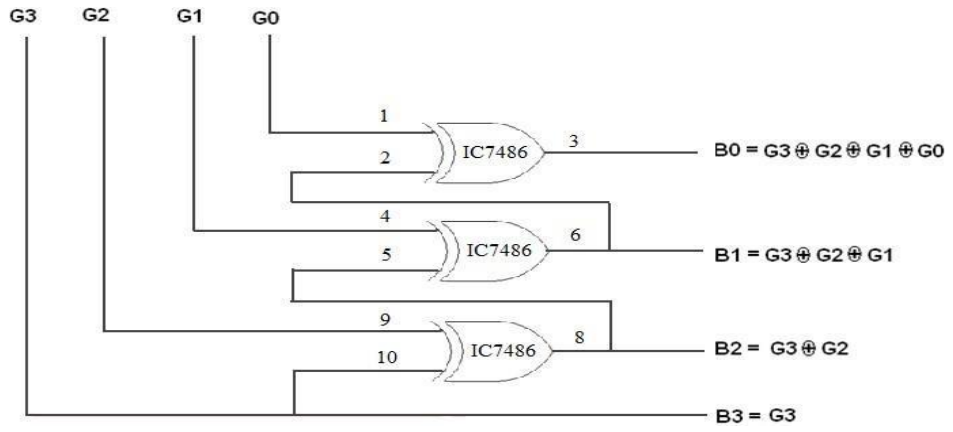
B1 = G3⊕G2⊕G1

K-Map for B0

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

B0 = G3⊕G2⊕G1⊕G0

LOGIC DIAGRAM FOR GRAY TO BINARY CODE CONVERTER:



RESULT:

Thus, the design for the code converter was done and verified using logic gates successfully.

DESIGN AND IMPLEMENTATION OF HALF ADDER AND FULL ADDER USING LOGIC GATES

DATE:

AIM:

To design and construct half adder, and full adder 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
4.	OR GATE	IC 7432	1
5.	IC TRAINER KIT	-	1
6.	CONNECTING WIRES	-	23

PROCEDURE:

a. Connections are given as per the logic diagram
b. Input are given to the circuit making high '1' i.e. +5 or Vcc Supply to the 14th pin and for low '0' i.e. GND to the 7th pin of the Gate IC
c. Depending upon the truth table, if the LED Glow it represents 1 and else it represents '0'
d. Verify the truth table as given

HALF ADDER: TRUTH TABLE:

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

K-Map for SUM:

A \ B	0	1
0		1
1	1	

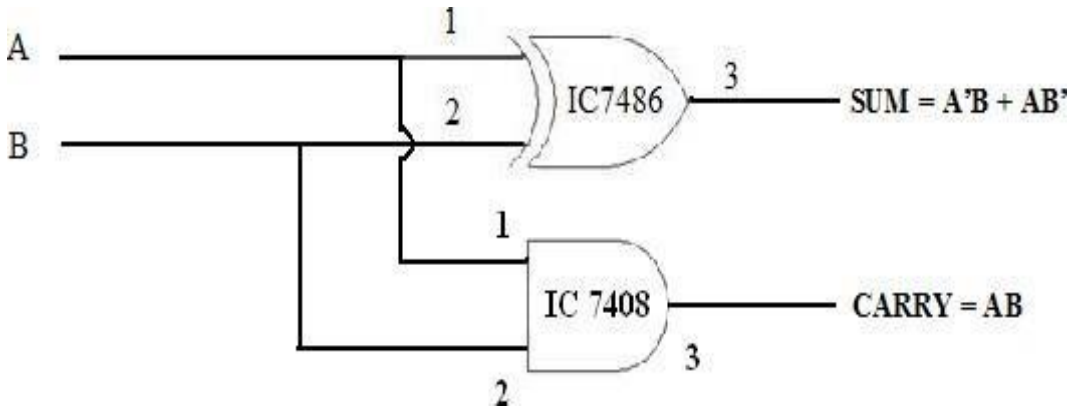
SUM = A'B + AB' = A ⊕ B

K-Map for CARRY:

A \ B	0	1
0		
1		1

CARRY = AB

LOGIC DIAGRAM FOR HALF ADDER:



FULL ADDER: TRUTH TABLE:

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

K-Map for SUM

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

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

$$= A \oplus B \oplus C$$

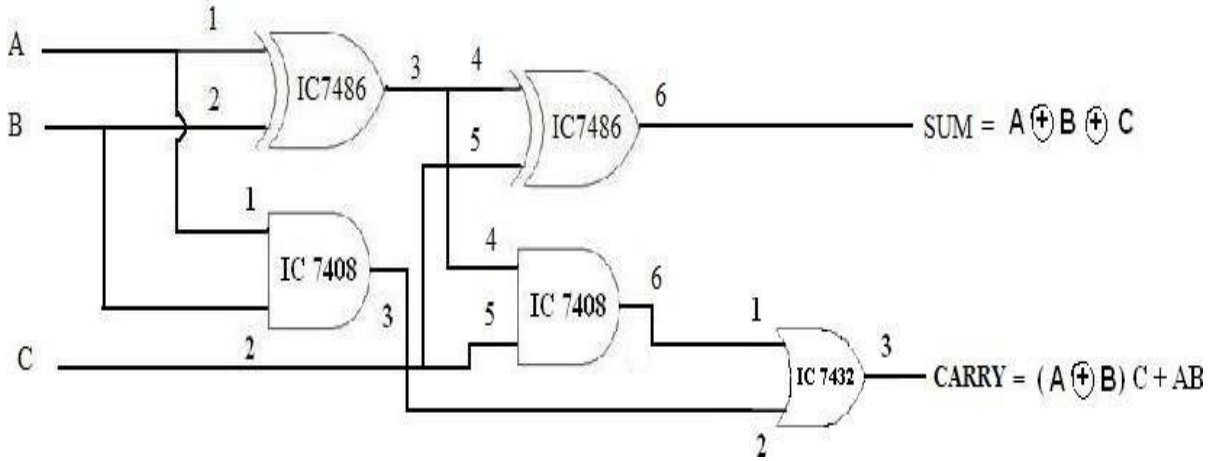
K-Map for CARRY

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

$$CARRY = A'BC + AB'C + ABC' + ABC$$

$$= (A \oplus B)\,C + AB$$

LOGIC DIAGRAM FOR FULL ADDER:



RESULT:

Thus, the design for half adder and full adder were designed and verified successfully using Truth table.

DESIGN AND IMPLEMENTATION OF HALF SUBTRACTOR AND FULL SUBTRACTOR USING LOGIC GATES

DATE:

AIM:

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

APPARATUS REQUIRED:

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

PROCEDURE:

a. Connections are given as per the logic diagram
b. Input is given to the circuit making high ‘1’ i.e. +5 or Vcc Supply to the 14th pin andfor low ‘0’ i.e. GND to the 7th pin of the Gate IC
c. Depending upon the truth table, if the LED Glow it represents 1 and else it represents ‘0’
d. Verify the truth table as given

HALF SUBRACTOR: TRUTH TABLE

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

K-Map for DIFFERENCE

A \ B	0	1
0		1
1	1	

DIFFERENCE = A'B + AB'

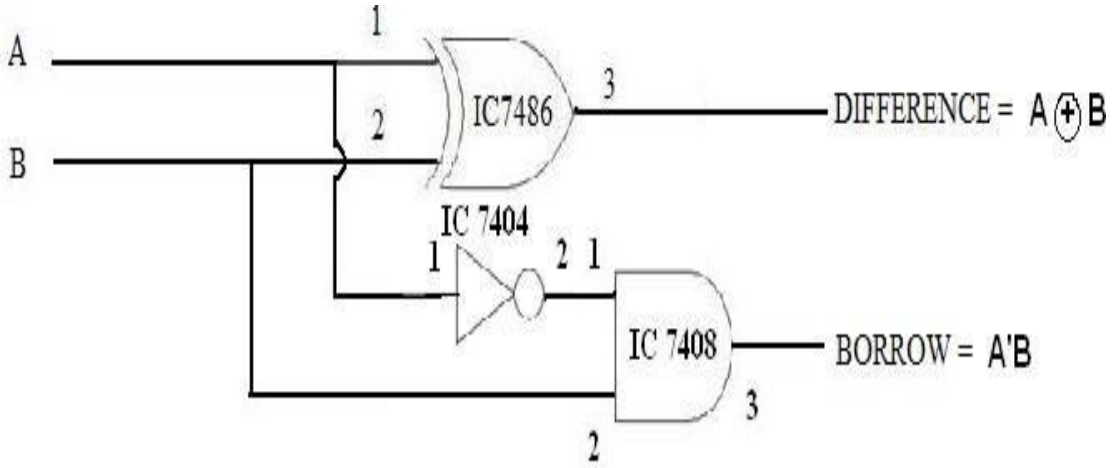
= A ⊕ B

K-Map for BORROW

A \ B	0	1
0		1
1		

BORROW = A'B

LOGIC DIAGRAM FOR HALF SUBRACTOR:



FULL SUBRACTOR: TRUTH TABLE:

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

K-Map for DIFFERENCE

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

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

= A ⊕ B ⊕ C

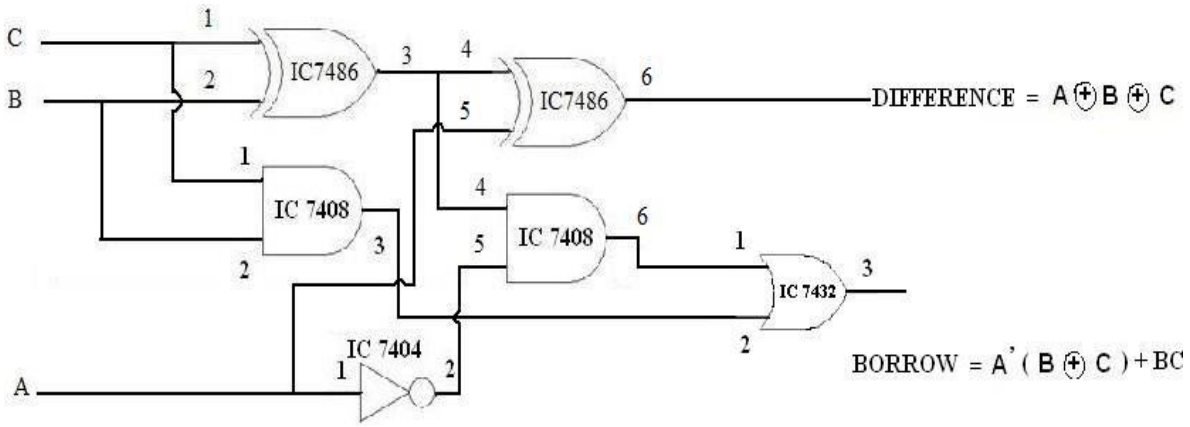
K-Map for BORROW

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

Borrow = A'B + BC + A'C

= A' (B ⊕ C) + BC

LOGIC DIAGRAM FOR FULL SUBTRACTOR:



RESULT:

Thus, the design for half subtractor and full subtractor were designed and verified successfully using the Truth table.

DESIGN AND IMPLEMENTATION OF MULTIPLEXERS

DATE:

AIM:

To Design and implement a 2 to 1 Multiplexer using logic gates.

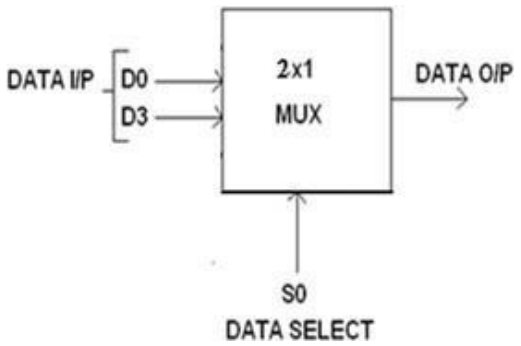
APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	AND GATE	IC 7408	1
2.	OR GATE	IC 7432	1
3.	NOT GATE	IC 7404	1
4.	IC TRAINER KIT	-	1
5.	CONNCECTING WIRES	-	32

PROCEDURE:

a. Connections are given as per the logic diagram
b. Input is given to the circuit making high '1' i.e. +5 or Vcc Supply to the 16th pin andfor low '0' i.e. GND to the 8th pin of the Gate IC
c. Depending upon the truth table, if the LED Glow represents 1 and else it represents '0'
d. Verify the truth table as given.

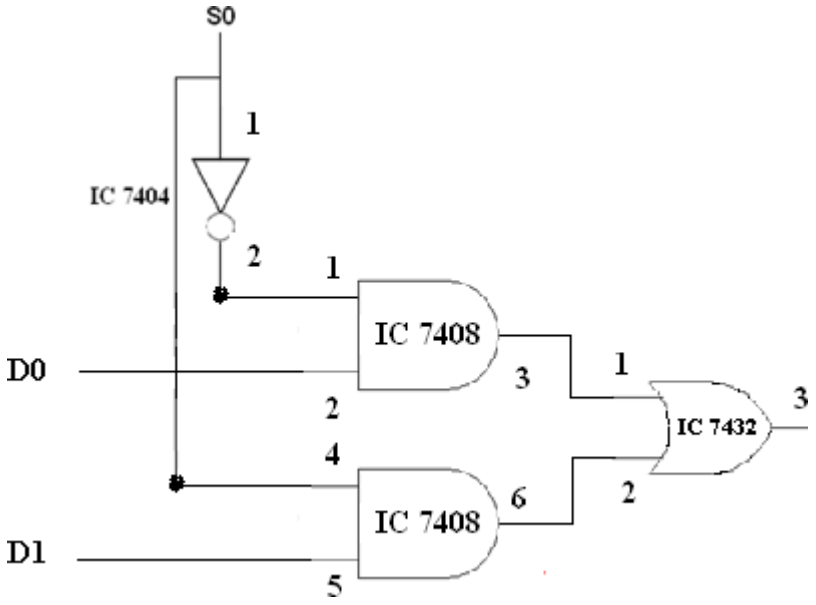
BLOCK DIAGRAM FOR 2:1 MULTIPLEXER:



MULTIPLEXER USING GATES:TRUTH TABLE:

S	Y =
0	D0
1	D1

CIRCUIT DIAGRAM FOR MULTIPLEXER:



RESULT:

Thus, the 2 to 1 Multiplexer circuit was designed using logic gates, and outputs are verified using the Truth table.

DESIGN AND IMPLEMENTATION OF DECODER

DATE:

AIM:

To design and implement a 2 to 4 Decoder using logic gates.

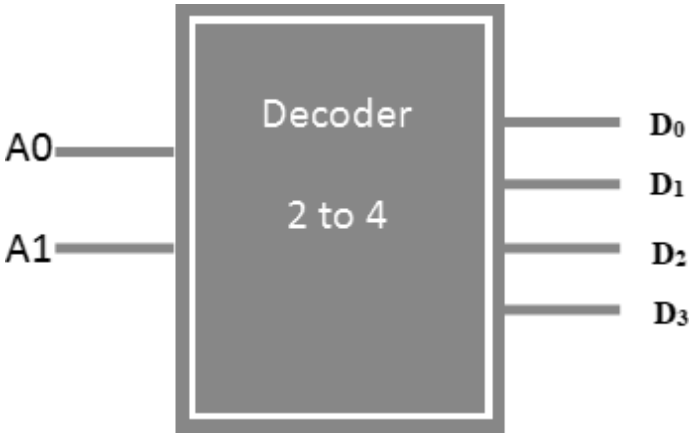
APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	AND GATE	IC 7408	1
2.	NOT GATE	IC7404	1
3.	IC TRAINER KIT	-	1
4.	CONNCECTING WIRES	-	few

PROCEDURE:

a. Connections are given as per the logic diagram
b. Input is given to the circuit making high '1' i.e. +5 or Vcc Supply to the 16th pin and for low '0' i.e. GND to the 8th pin of the Gate IC
c. Depending upon the truth table, if the LED Glow it represents 1 and else it represents '0'
d. Verify the truth table as given

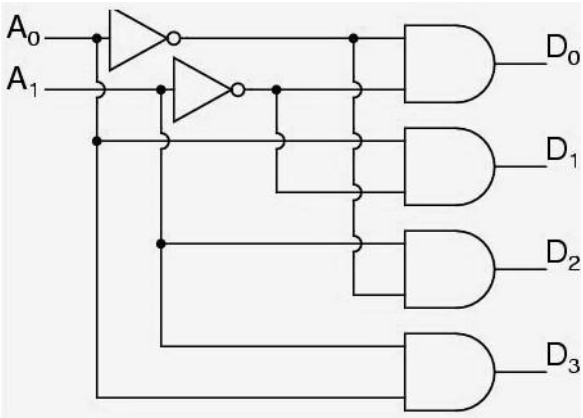
LOCK DIAGRAM FOR 2:4 DECODER:



TRUTH TABLE:

INPUT		OUTPUT			
A1	**A0**	**D0**	**D1**	**D2**	**D3**
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	0	0	0	0	1

CIRCUIT DIAGRAM FOR DECODER:



RESULT:

Thus, the 2 to 4 Decoder circuit was designed using logic gates and outputs are verified using Truth table.

DESIGN AND IMPLEMENTATION OF FLIP-FLOPS

DATE:

AIM:

To Design and implement Flipflops (D, SR, JK & T) using logic gates.

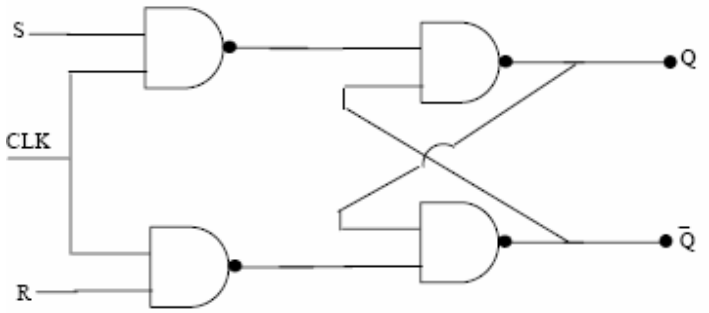
APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	AND GATE	IC 7408	1
2.	OR GATE	IC 7432	1
3.	NOT GATE	IC 7404	1
4.	IC TRAINER KIT	-	1
5.	CONNCECTING WIRES	-	32

PROCEDURE:

a. Connections are given as per the logic diagram
b. Input are given to the circuit making high '1' i.e. +5 or Vcc Supply to the 16th pin and for low '0' i.e. GND to the 8th pin of the Gate IC
c. Depending upon the truth table, if the LED Glow it represents 1 and else it represents '0'
d. Verify the truth table as given

SR FLIPFLOP
LOGIC DIAGRAM:

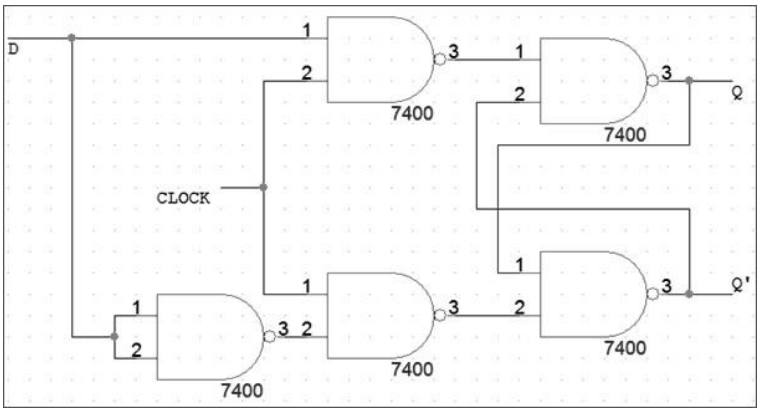


TRUTH TABLE:

Inputs			Outputs
R	S	CLK	Qt+1
0	0	↑	Qt (Previous State)
0	1	↑	1
1	0	↑	0
1	1	↑	Indeterminate state

D FLIPFLOP

LOGIC DIAGRAM:

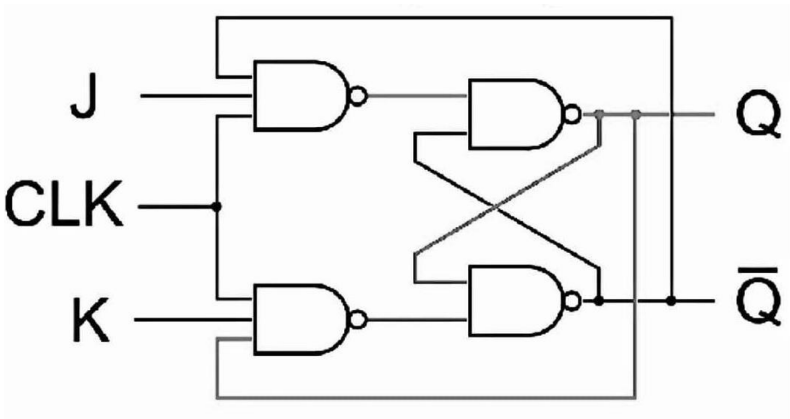


TRUTH TABLE:

Outputs		
D	CLK	Q
0	↑	0
1	↑	1

JK FLIPFLOP

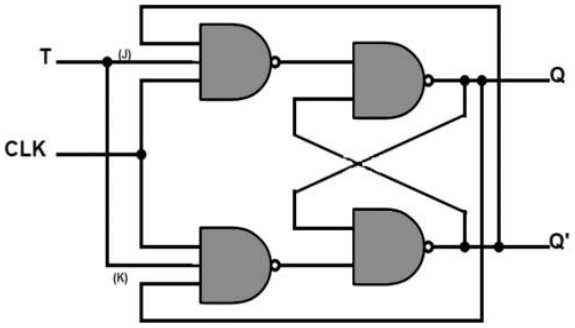
LOGIC DIAGRAM:



TRUTH TABLE:

Truth Table			
Q	**J**	**K**	**Q(t+1)**
0	x	x	Qt (PreviousState)
1	0	0	Qt (PreviousState)
1	0	1	0
1	1	0	1
1	1	1	Qt'(Toggle)

T FLIPFLOP LOGIC DIAGRAM:



TRUTH TABLE:

Q	T	Q(t+1)
0	x	Qt (Previous State)
1	0	Qt (Previous State)
1	1	Qt'(Toggle)

RESULT

Thus, the Flip flops (D, SR, JK & T) using logic gates were Designed and verified the truth tables.

DESIGN OF 2 – BIT SYNCHRONOUS COUNTER

DATE:

AIM:

To Design and implement 2 – bit synchronous counter using Flip-Flops.

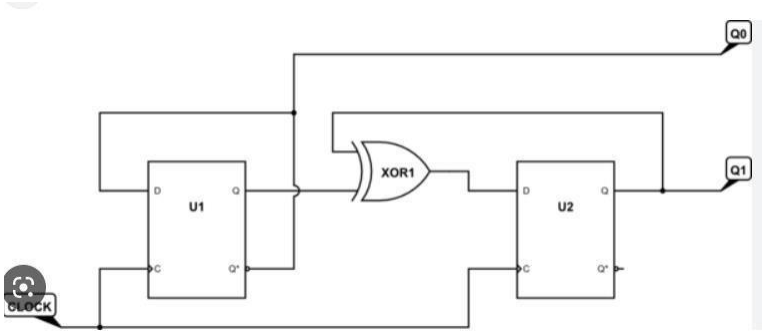
APPARATUS REQUIRED:

Sl. No	COMPONENT	SPECIFICATION	QTY.
1.	D FF	IC 7474	1
2.	OR GATE	IC 7432	1
3.	NOT GATE	IC 7404	1
4.	IC TRAINER KIT	-	1
5.	CONNECTING WIRES	-	few

PROCEDURE:

a. Connections are given as per the logic diagram
b. Input is given to the circuit making high '1' i.e. +5 or Vcc Supply to the 16th pin andfor low '0' i.e. GND to the 8th pin of the Gate IC
c. Depending upon the truth table, if the LED Glow it represents 1 and else it represents '0'
d. Verify the truth table as given

LOGIC DIAGRAM:



TRUTH TABLE:

CLK	QA	QB
1	0	0
2	0	1
3	1	0
4	1	1

RESULT:

The 2-Bit counter sequential circuit was designed, and implemented and truth tablewas verified.

DESIGN AND IMPLEMENTATION OF SHIFT REGISTERS

DATE:

AIM:

To design and implement

A. Serial in serial out
B. Serial in parallel out

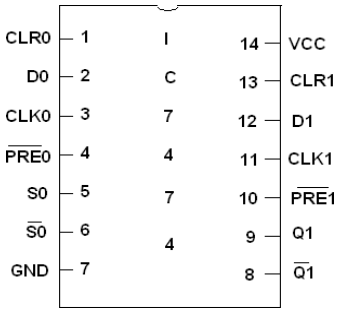
APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	D FLIP FLOP	IC 7474	2
2.	IC TRAINER KIT	-	1
3.	CONNECTING WIRES	-	35

PROCEDURE:

a. Connections are given as per the logic diagram
b. Input is given to the circuit making high '1' i.e., +5 or Vcc Supply to the 14th pin and for low '0' i.e., GND to the 7th pin of the Gate IC
c. Depending upon the truth table, if the LED Glow it represents 1 and else it represents '0'
d. Verify the truth table as given

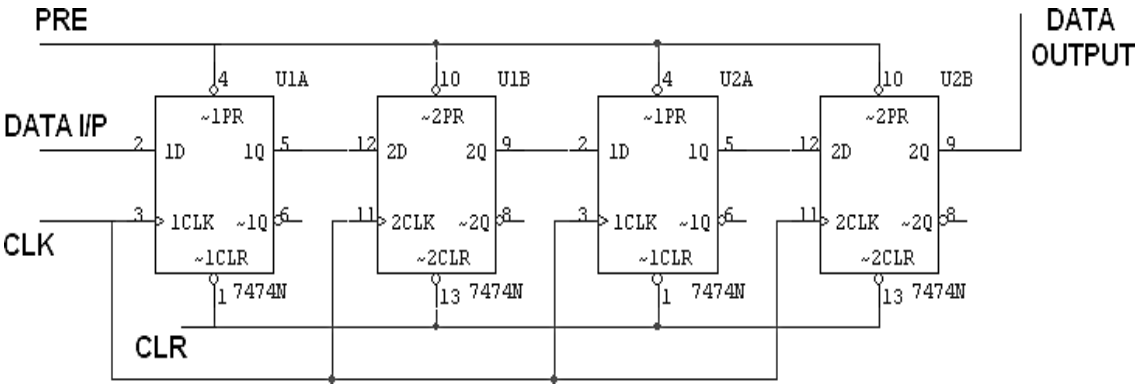
PIN DIAGRAM:



SERIAL IN SERIAL OUTTRUTH TABLE:

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

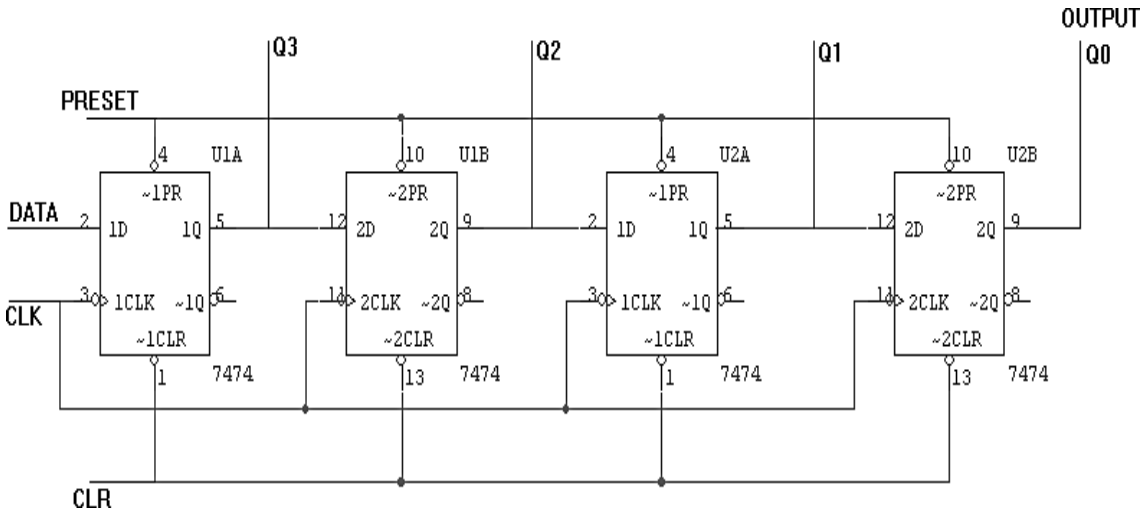
LOGIC DIAGRAM:



SERIAL IN PARALLEL OUTTRUTH 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	0
4	1	1	0	0	1

LOGIC DAIGRAM:



RESULT:

Thus, the shift register using D Flip Flop is constructed and the outputs are verified using truth table.

SIMULATION OF LOGIC GATES USING VHDL

DATE:

AIM:

To write the source code and the simulation of the logic gates of AND, OR and NOT using VHDL.

SOFTWARE REQUIRED:

ModelSim Simulator

PROCEDURE:

Step1: Define the specifications and initialize the design.
Step2: Declare the name of the module by using source code.
Step3: Write the source code in VHDL.
Step4: Check the syntax and debug the errors if found, obtain the synthesis is report.
Step5: Verify the output by simulating the source code.

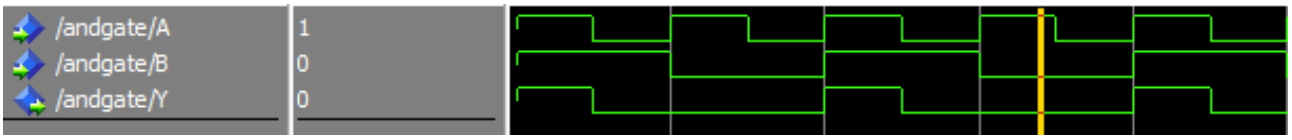
LOGIC GATES:

AND GATE:

VHDL CODE:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity andGate is
port(A : in std_logic;      -- AND gate input B : in std_logic;      -- AND gate input Y : out std_logic); -
- AND gate output
end andGate;
architecture andLogic of andGate isbegin
        Y <= A AND B;
end andLogic;
```

OUTPUT WAVEFORM:



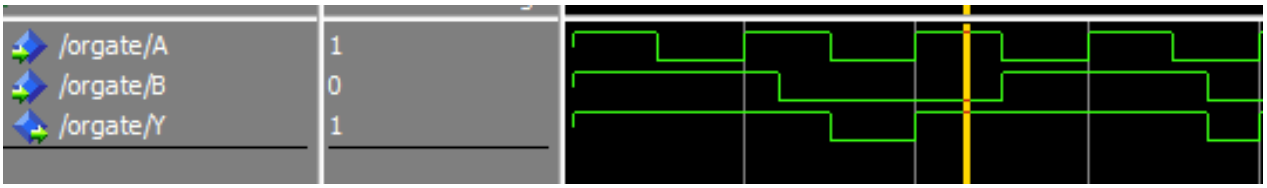
OR GATE:

VHDL CODE:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity orGate is

        port(A : in std_logic;        -- OR gate input B : in std_logic;
                -- OR gate input Y : out std_logic); -- OR gate output
end orGate;
architecture orLogic of orGate isbegin
        Y <= A OR B;
end orLogic;
```

OUTPUT:

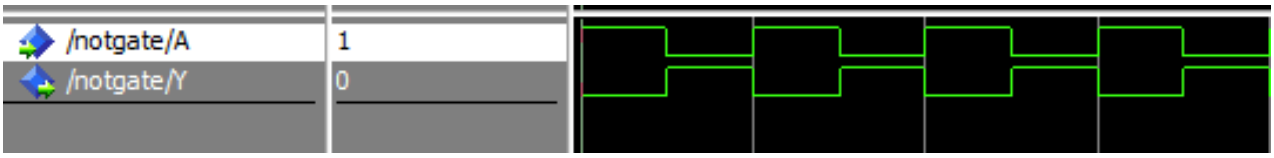


NOT GATE:

VHDL CODE:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity notGate is
port(A : in std_logic;          -- NOT gate input
Y : out std_logic);  -- NOT gate output
end notGate;
architecture notLogic of notGate isbegin
        Y <= not A;
end notLogic;
```

OUTPUT WAVEFORM:



RESULT

Thus, the VHDL code for logic gates were written, executed and verified the output.

SIMULATION OF ADDER CIRCUITS USING VHDL

DATE:

AIM:

To write the source code and the simulate the Half adder and Full adder using VHDL.

SOFTWARE REQUIRED:

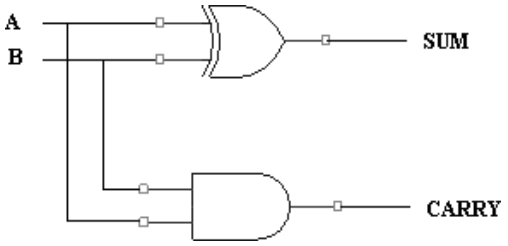
ModelSim Simulator

PROCEDURE:

Step1: Define the specifications and initialize the design.
Step2: Declare the name of the module by using source code.Step3: Write the source code in VHDL.
Step4: Check the syntax and debug the errors if found, obtain the synthesis is report.Step5: Verify the output by simulating the source code.

HALF ADDER:

LOGIC DIAGRAM:



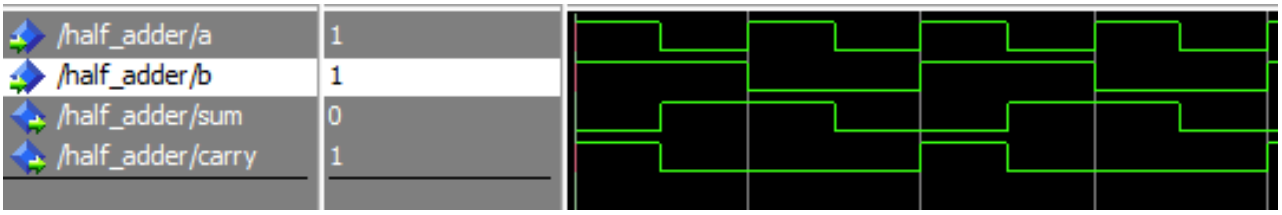
TRUTH TABLE:

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

VHDL CODE:

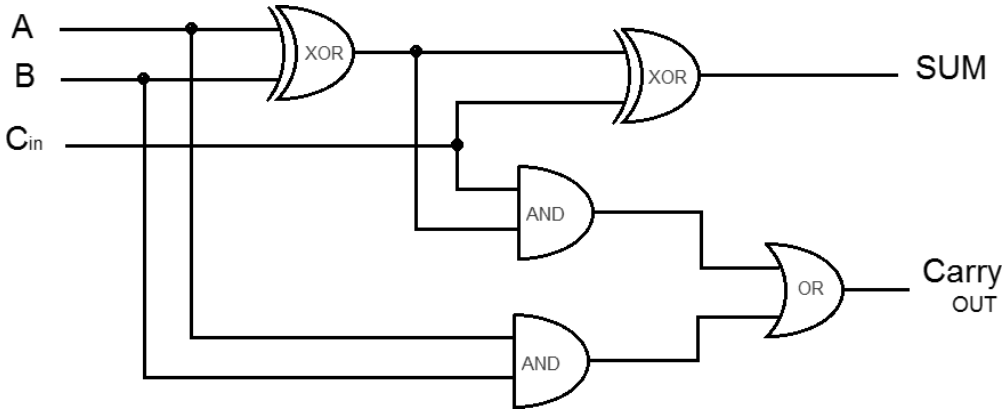
```
Library ieee;
use ieee.std_logic_1164.all;
entity half_adder is
 port(a,b:in bit; sum,carry:out bit);end half_adder;
 architecture data of half_adder isbegin
sum<= a xor b; carry <= a
and b;
        end data;
```

OUTPUT:



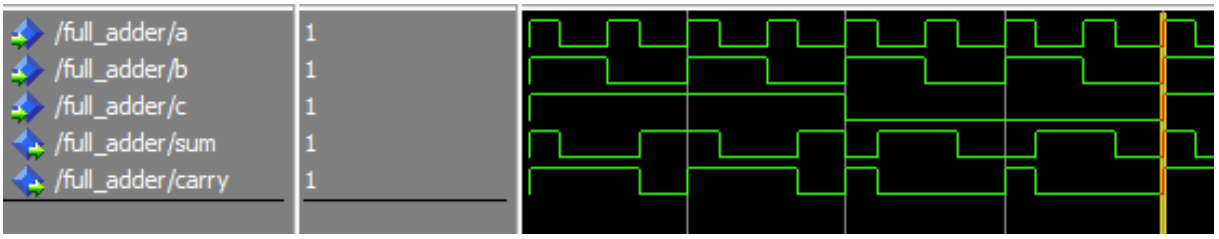
FULL ADDER Circuit Diagram:

VHDL CODE:




```
Library ieee;
use ieee.std_logic_1164.all;
entity full_adder is port (a,b,c:in bit; sum,carry:out bit);end
full_adder;
architecture data of full_adder isbegin
sum<= a xor b xor c;
carry <= ((a and b) or (b and c) or (a and c));end data;
```

Output;



RESULT

Thus, the VHDL code for half adder and full adder were written, executed and verified the output.

SIMULATION OF SUBTRACTOR CIRCUITS USING VHDL

DATE:

AIM:

To write the source code and simulate the Half subtractor and Full subtractor using VHDL.

SOFTWARE REQUIRED:

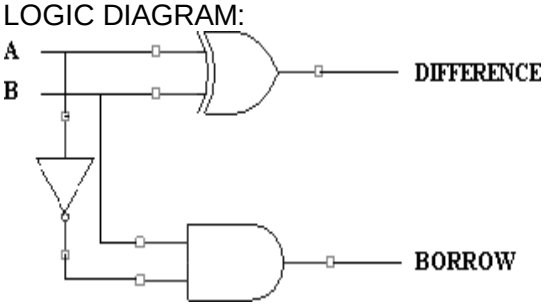
ModelSim Simulator

PROCEDURE:

Step1: Define the specifications and initialize the design.
Step2: Declare the name of the module by using source code.
Step3: Write the source code in VHDL.
Step4: Check the syntax and debug the errors if found, obtain the synthesis is report.

HALF SUBTRACTOR:

LOGIC DIAGRAM:



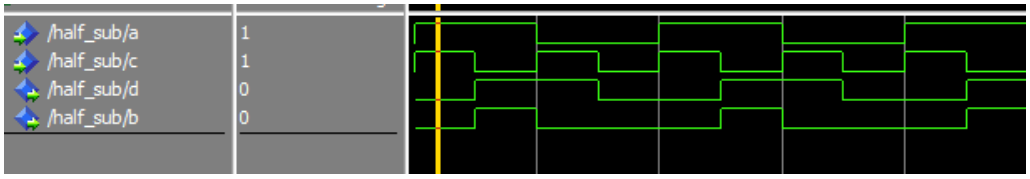
TRUTH TABLE:

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

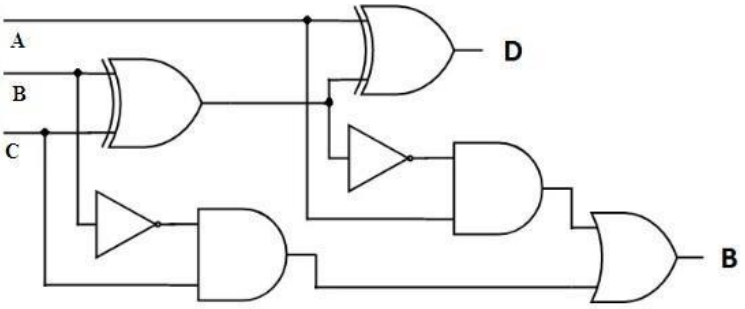
VHDL CODE:

```
Library ieee;
use ieee.std_logic_1164.all; entity half_sub
is port(a,c:in bit; d,b:out bit);
end half_sub;
architecture data of half_sub isbegin
d<= a xor c;
   b<= (a and (not c));end data;
```

OUTPUT:



FULL SUBTRACTOR CIRCUIT DIAGRAM:



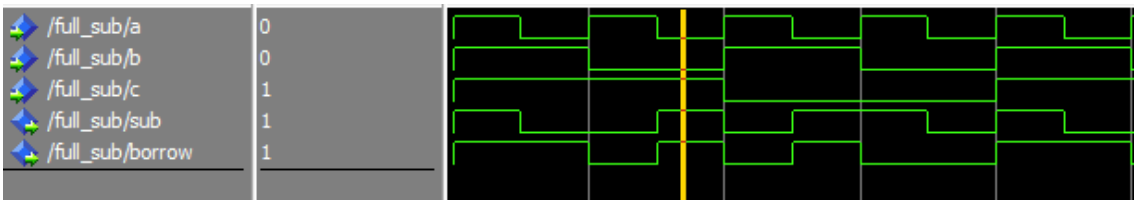
TRUTH TABLE

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

VHDL CODE:

```
 Library ieee;
use ieee.std_logic_1164.all;entity
 full_sub is
   port (a,b,c:in bit; sub,borrow:out bit);end
 full_sub;
  architecture data of full_sub isbegin
 sub<= a xor b xor c;
   borrow <= ((b xor c) and (not a)) or (b and c);end
 data;
```

Output:



RESULT

Thus, the VHDL code for half subtractor and full subtractor were written, executed andverified the output.

SIMULATION OF MUTIPLEXER AND DEMULTIPLEXER USING VHDL

DATE:

AIM:

To write the source code and the simulate the 4x1 Multiplexer and 1x4 Demultiplexerusing VHDL.

SOFTWARE REQUIRED:

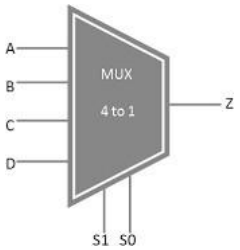
ModelSim Simulator

PROCEDURE:

Step1: Define the specifications and initialize the design.
Step2: Declare the name of the module by using source code.Step3: Write the source code in VHDL.
Step4: Check the syntax and debug the errors if found, obtain the synthesis is report.Step5: Verify the output by simulating the source code.

1x4 MUTIPLEXER:

LOGIC DIAGRAM:



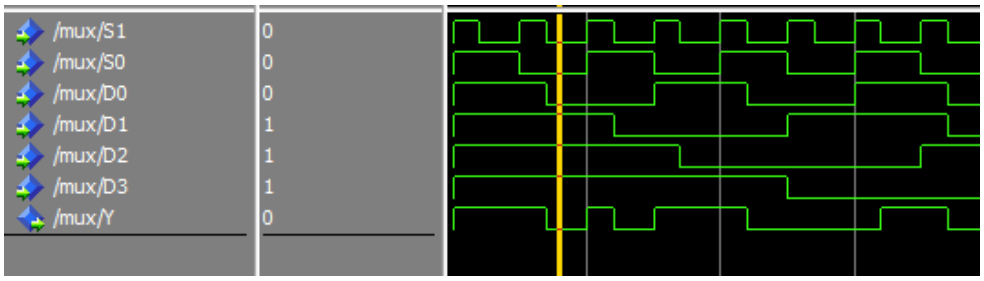
TRUTH TABLE:

S1	S0	Output
0	0	A
0	1	B

VHDL CODE:

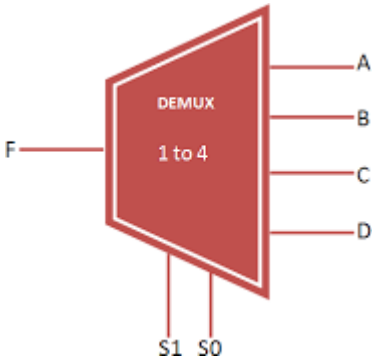
```
Library ieee;
use
  ieee.std_logic_1164.all;
  entity mux is
   port(S1,S0,D0,D1,D2,D3:in bit; Y:out bit);
end mux;
  architecture data of mux
isbegin
   Y<= (not S0 and not S1 and D0) or
      (S0 and not S1 and D1) or
      (not S0 and S1 and D2)
      or(S0 and S1 and D3);
end data;
```

OUTPUT:


/mux/S1
/mux/S0
/mux/D0
/mux/D1
/mux/D2
/mux/D3
/mux/Y
0
0
0
1
1
1
0

4x1 DEMUTIPLEXER:

LOGIC DIAGRAM:

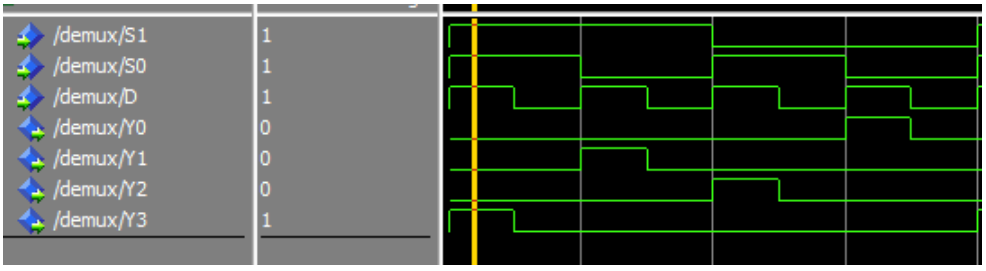


INPUT			OUTPUT			
D	S0	S1	Y0	Y1	Y2	Y3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

VHDL CODE:

```
Library ieee;
use
  ieee.std_logic_1164.all;
  entity demux is
   port(S1,S0,D:in bit; Y0,Y1,Y2,Y3:out bit);
end demux;
  architecture data of demux
isbegin
   Y0<= ((Not S0) and (Not S1) and D);
   Y1<=  ((Not S0) and S1 and D);
   Y2<=  (S0 and (Not S1) and D);
   Y3<=  (S0 and S1 and D);
end data;
```

Output:



RESULT

Thus, the VHDL code for Multiplexer and Demultiplexer were written, executed and verified the output.

SIMULATION OF ENCODER AND DECODER USING VHDL

DATE:

AIM:

To write the source code and simulate the encoder and decoder using VHDL.

SOFTWARE REQUIRED:

ModelSim Simulator

PROCEDURE:

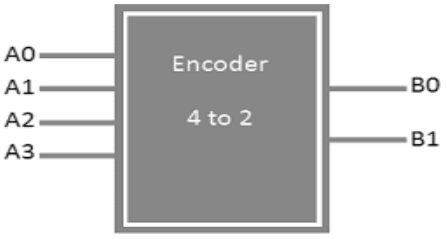
Step1: Define the specifications and initialize the design.
Step2: Declare the name of the module by using source code.Step3: Write the source code in VHDL.
Step4: Check the syntax and debug the errors if found, obtain the synthesis is report.Step5: Verify the output by simulating the source code.

4x2 ENCODER:

TRUTH TABLE:

INPUT				OUTPUT	
A3	A2	A1	A0	B1	B0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

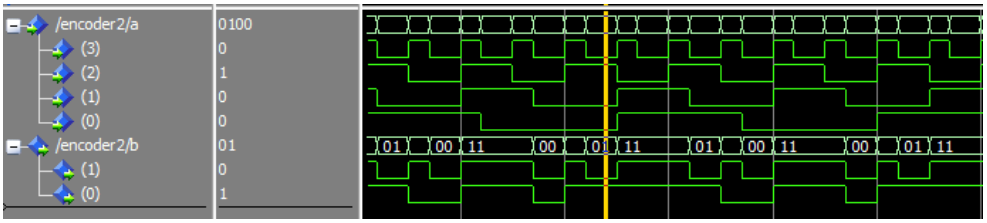
LOGIC DIAGRAM:



VHDL CODE:

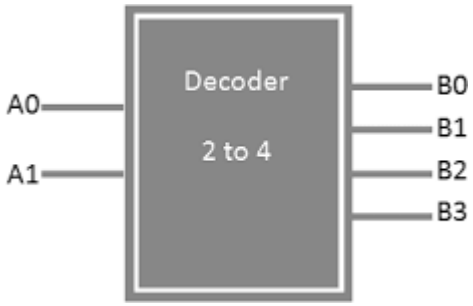
```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity encoder2 is
 port(
 a : in STD_LOGIC_VECTOR(3 downto 0); b
 : out STD_LOGIC_VECTOR(1 downto 0)
 );
end encoder2;
architecture bhv of encoder2
isbegin
b(0) <= a(1) or a(2);
b(1) <= a(1) or a(3);
end bhv;
```

OUTPUT:



4x2 DECODER:

LOGIC DIAGRAM:



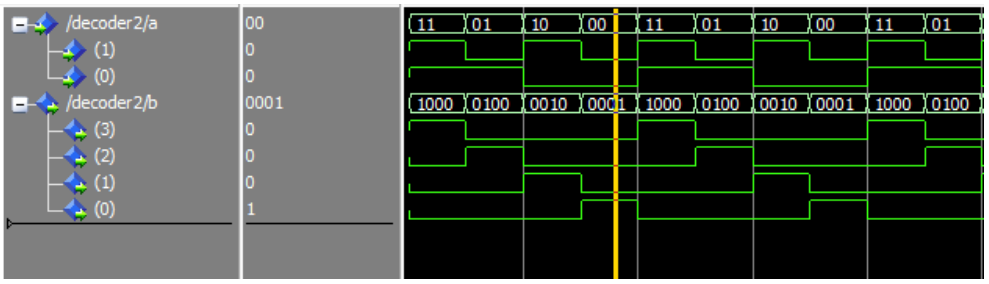
TRUTH TABLE:

INPUT		OUTPUT			
A1	A0	B3	B2	B1	B0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

VHDL CODE:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity decoder2 isport( a : in STD_LOGIC_VECTOR(1 downto
 0); b : out STD_LOGIC_VECTOR(3 downto 0));
end decoder2;
architecture bhv of decoder2 is
begin
b(0) <= not a(0) and not a(1);
b(1) <= not a(0) and a(1);
b(2) <= a(0) and not a(1);
b(3) <= a(0) and a(1);
end bhv;
```

OUTPUT:



RESULT

Thus, the VHDL code for Encoder and Decoder were written, executed and verified the output.