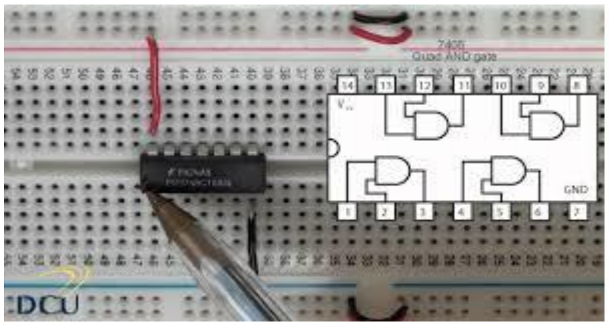
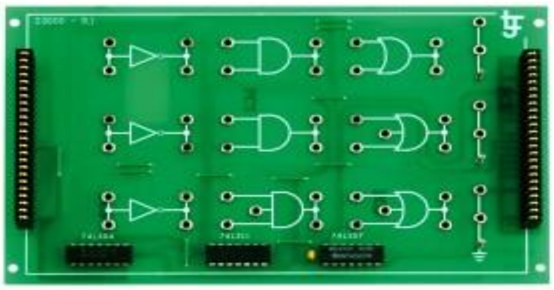
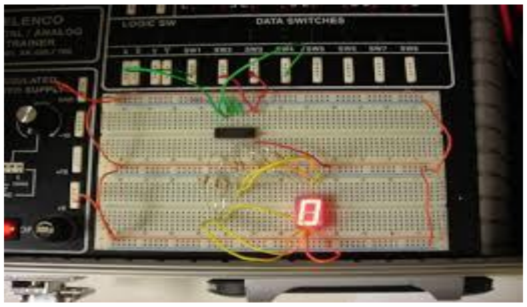
LABORATORY MANUAL

Digital Electronics Laboratory







Department of Instrumentation Engineering

JORHAT ENGINEERING COLLEGE

Assam-785007

Experiment No.	**Title of the Experiment**	**Objective of the Experiment**
1	To study and verify the truth table of logic gates	Identify various ICs and their specification a. OR gate b. AND gate c. NAND gate d. NOR gate
2	Realization of a Boolean function	To simplify the given expression and to realize it using Basic gates and Universal gate
3	Design and implementation using NAND gate	To realize why NAND gate is known as the universal gate by implementation of : a. NOT using NAND b. AND using NAND c. OR using NAND d. XOR using NAND
4	Adders and Subtractors	To realize a. Half Adder and Full Adder b. Half Subtractor and Full Subtractor by using Basic gates and NAND gates
5	Binary to grey generator	To learn the importance of weighted and non weighted code To learn to generate gray code .
6	Multiplexer and Demultiplexer	a. To design and set up a 4:1 Multiplexer (MUX) using only NAND gates. b. To design and set up a 1:4 Demultiplexer(DE-MUX) using only NAND gates.
7	Realization of a Boolean function using Logisim	To learn the use of Logisim software to design digital electronics circuits.
8	Flip Flop	a. Truth Table verification of 1) RS Flip Flop 2) T type Flip Flop. 3) D type Flip Flop. 4) JK Flip Flop. b. Conversion of one type of Flip flop to another

Sl. No.	Title of the Experiment	Remarks
1	To study and verify the truth table of logic gates	
2	Realization of a Boolean function	
3	Design and implementation using NAND gate	
4	Adders and Subtractors	
5	Binary to grey generator	
6	Multiplexer and Demultiplexer	
7	Realization of a Boolean function using Logisim	
8	FlipFlop	

EXPERIMENT: 1 LOGIC GATES

AIM: To study and verify the truth table of logic gates

OBJECTIVE:

Identify various ICs and their specification

a. OR gate
b. AND gate
c. NAND gate
d. NOR gate

COMPONENTS REQUIRED:

- Breadboard.
- Connecting wires.
- IC 7400, IC 7408, IC 7432, IC 7406, IC 7402, IC 7404, IC 7486

THEORY:

The basic logic gates are the building blocks of more complex logic circuits. These logic gates perform the basic Boolean functions, such as AND, OR, NAND, NOR, Inversion, Exclusive-OR, Exclusive-NOR. Fig. below shows the circuit symbol, Boolean function, and truth. It is seen from the Fig that each gate has one or two binary inputs, A and B, and one binary output, C. The small circle on the output of the circuit symbols designates the logic complement. The AND, OR, NAND, and NOR gates can be extended to have more than two inputs. A gate can be extended to have multiple inputs if the binary operation it represents is commutative and associative.

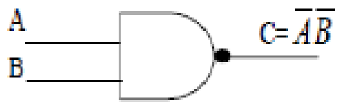
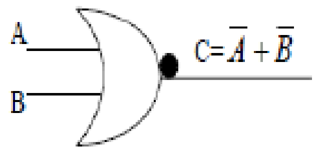
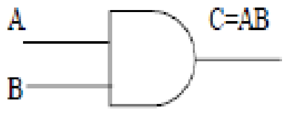
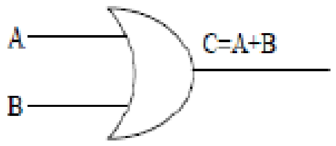
These basic logic gates are implemented as small-scale integrated circuits (SSICs) or as part of more complex medium scale (MSI) or very large-scale (VLSI) integrated circuits. Digital IC gates are classified not only by their logic operation, but also the specific logic-circuit family to which they belong. Each logic family has its own basic electronic circuit upon which more complex digital circuits and functions are developed. The following logic families are the most frequently used.

TTL	Transistor-transistor logic
ECL	Emitter-coupled logic
MOS	Metal-oxide semiconductor
CMOS	Complementary metal-oxide semiconductor

TTL and ECL are based upon bipolar transistors. TTL has a popularity among logic families. ECL is used only in systems requiring high-speed operation. MOS and CMOS, are based on field effect transistors. They are widely used in large scale integrated circuits because of their high component density and relatively low power consumption. CMOS logic consumes far less power than MOS logic. There are various commercial integrated circuit chips available. TTL ICs are usually distinguished by numerical designation as the 5400 and 7400 series.

PROCEDURE:

1. Check the components for their working.
2. Insert the appropriate IC into the IC base.
3. Make connections as shown in the circuit diagram.
4. Provide the input data via the input switches and observe the output on output LEDs

S.NO	GATE	SYMBOL	INPUTS		OUTPUT
			A	B	C
1.	NAND IC 7400		0	0	1
			0	1	1
			1	0	1
			1	1	0
2.	NOR IC 7402		0	0	1
			0	1	0
			1	0	0
			1	1	0
3.	AND IC 7408		0	0	0
			0	1	0
			1	0	0
			1	1	1
4.	OR IC 7432		0	0	0
			0	1	1
			1	0	1
			1	1	1

QUESTIONS:

1. Why NAND & NOR gates are called universal gates?
2. Realize the EX – OR gates using minimum number of NAND gates.
3. Give the truth table for EX-NOR and realize using NAND gates?
4. What are the logic low and High levels of TTL IC’s and CMOS IC’s?
5. Compare TTL logic family with CMOS family?
6. Which logic family is fastest and which has low power dissipation?

EXPERIMENT: 2 REALIZATION OF A BOOLEAN FUNCTION.

AIM: To simplify the given expression and to realize it using Basic gates
and Universal gates

LEARNING OBJECTIVE:

- To simplify the Boolean expression and to build the logic circuit.
- Given a Truth table to derive the Boolean expressions and build the logic circuit to realize it.

COMPONENTS REQUIRED:

IC 7400, IC 7408, IC 7432, IC 7406, IC 7402, Connecting wires, Bread board.

THEORY:

Canonical Forms (Normal Forms): Any Boolean function can be written in disjunctivenormal form (sum of min-terms) or conjunctive normal form (product of max-terms).

A Boolean function can be represented by a Karnaugh map in which each cell corresponds to a minterm. The cells are arranged in such a way that any two immediately adjacent cells correspond to two minterms of distance 1. There is more than one way to construct a map with this property.

Karnaugh Maps

For a function of two variables, say, f(x, y),

	x'	x
y'	f(0,0)	f(1,0)
y	f(0,1)	f(1,1)

For a function of three variables, say, f(x, y, z)

	x'y'	x'y	xy	xy'
z'	f(0,0,0)	f(0,1,0)	f(1,1,0)	f(1,0,0)
z	f(0,0,1)	f(0,1,1)	f(1,1,1)	f(1,0,1)

For a function of four variables: f(w, x, y, z)

	w'x'	w'x	wx	wx'
y'z'	0	4	12	8
y'z	1	5	13	9
yz	3	7	15	11
yz'	2	6	14	10

Realization of Boolean expression:

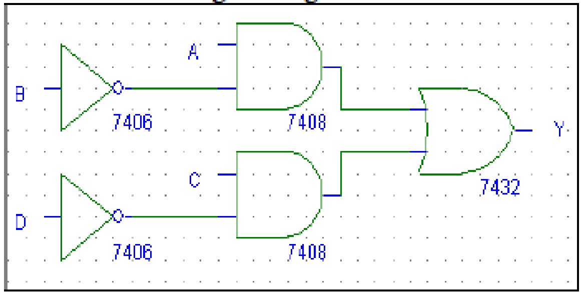
1) $Y = \bar{A}\bar{B}C\bar{D} + \bar{A}BC\bar{D} + ABC\bar{D} + A\bar{B}C\bar{D} + A\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}D + A\bar{B}CD$

AB

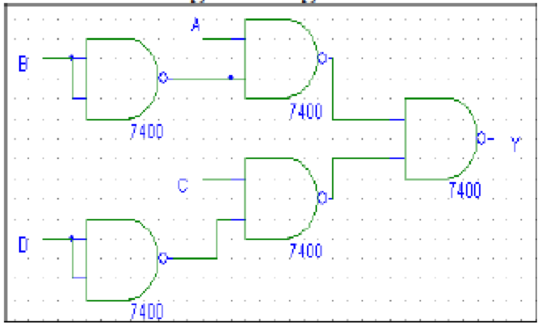
			1
			1
			1
1	1	1	1

After simplifying using K-Map method we get $Y = A\bar{B} + C\bar{D}$

Realization using Basic gates



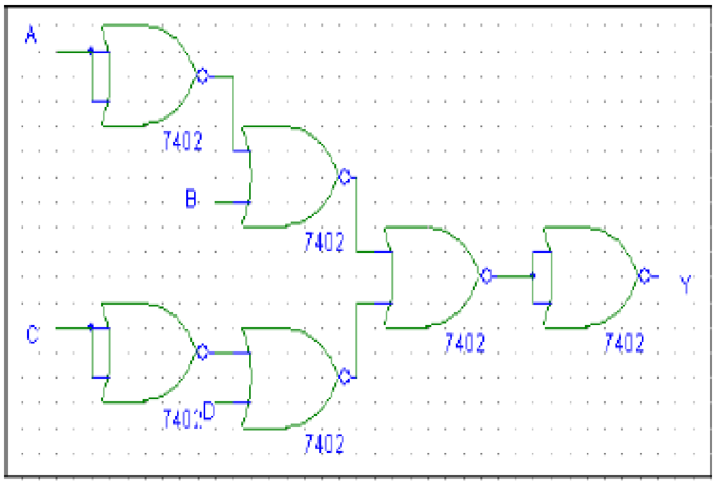
Realization using NAND gates



TRUTH TABLE

INPUTS				OUTPUT
A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

Realization using NOR gates



2) For the given Truth Table, realize a logical circuit using basic gates and NAND gates

Inputs				Output
A	**B**	**C**	**D**	**Y**
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

<u>PROCEDURE:</u>

Check the components for their working.

Insert the appropriate IC into the IC base.

Make connections as shown in the circuit diagram.

Provide the input data via the input switches and observe the output on output LEDs Verify the Truth Table

RESULT: Simplified and verified the Boolean function using basic gates and universal gates

QUESTIONS:

1) What are the different methods to obtain minimal expression?
2) Realize I(A,B,C,D) =∑(0,1,2,3,4,5,6,7,8,9,10) and show the truthtables.
3) What is a Min term and Max term
4) State the difference between SOP and POS.
5) What is meant by canonical representation?
6) What is K-map? Why is it used?
7) What are universal gates?

EXPERIMENT: 3 DESIGNS AND IMPLEMENTATION USING NAND GATE

AIM: To design and implementation using NAND gate

OBJECTIVE:

- To realize why NAND gate is known as the universal gate by implementation of :

a. NOT using NAND
b. AND using NAND
c. OR using NAND
d. XOR using NAND

COMPONENTS REQUIRED:

- Breadboard.
- Connecting wires.
- IC 7400, IC 7408, IC 7432, IC 7406, IC 7402, IC 7404, IC 7486

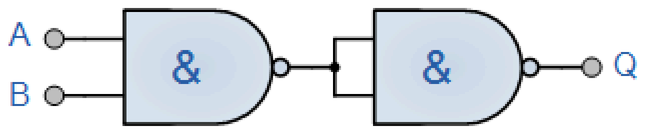
THEORY:

Boolean algebra is a branch of mathematical logic, where the variables are either true (1) or false (0).In order to construct NOT, AND, OR, XOR gates from NAND gates only, we need to be familiar with the following boolean algebra laws:

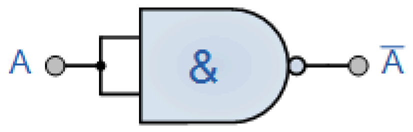
1. Involution Law

2. Idempotency (Idempotent) law

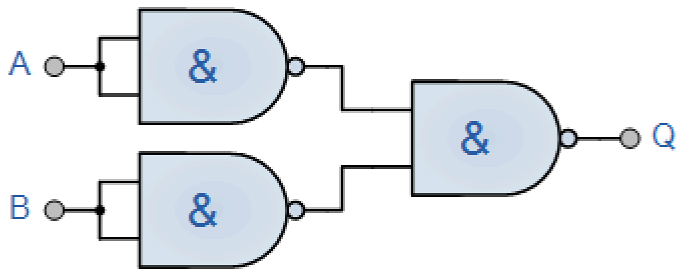
3. DeMorgan's Law



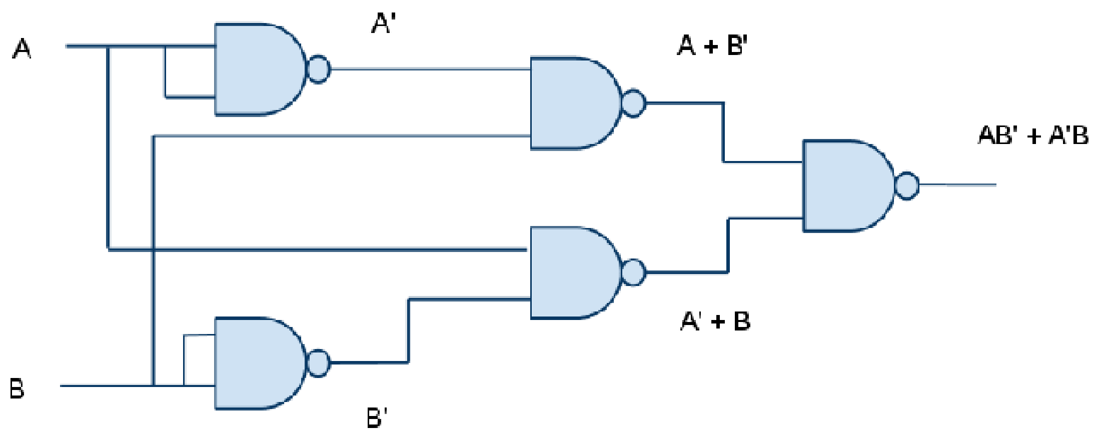
2-input AND Gate



NOT Gate



2-input OR Gate



2 input XOR gate

PROCEDURE:

Check the components for their working.

Insert the appropriate IC into the breadboard.

Make connections as shown in the circuit diagram.

Provide the input data via the input switches and observe the output on output LEDs

Verify the Truth Table

RESULT:

NOT, AND, OR, XOR gate is realized using NAND gate.

EXPERIMENT:4 ADDERS AND SUBTRACTORS

AIM: To realize

i) Half Adder and Full Adder

ii) Half Subtractor and Full Subtractor by using Basic gates and NAND gates

LEARNING OBJECTIVE:

- To realize the adder and subtractor circuits using basic gates and universal gates
- To realize full adder using two half adders
- To realize a full subtractor using two half subtractors

COMPONENTS REQUIRED:

IC 7400, IC 7408, IC 7486, IC 7432, Patch Cords & IC Trainer Kit.

THEORY:

Half-Adder: A combinational logic circuit that performs the addition of two data bits, A and B, is called a half-adder. Addition will result in two output bits; one of which is the sum bit, S, and the other is the carry bit, C. The Boolean functions describing the half-adder are:

$S = A \oplus B$ $\qquad$ $C = A B$

Full-Adder: The half-adder does not take the carry bit from its previous stage into account. This carry bit from its previous stage is called carry-in bit. A combinational logic circuit that adds two data bits, A and B, and a carry-in bit, Cin , is called a full-adder. The Boolean functions describing the full-adder are:

$S = (x \oplus y) \oplus Cin$ $\qquad$ $C = xy + Cin\,(x \oplus y)$

Half Subtractor: Subtracting a single-bit binary value B from another A (i.e. A -B) produces a difference bit D and a borrow out bit B-out. This operation is called half subtraction and the circuit to realize it is called a half subtractor. The Boolean functions describing the half-Subtractor are:

$S = A \oplus B$ $\qquad$ $C = A' B$

Full Subtractor: Subtracting two single-bit binary values, B, Cin from a single-bit value A produces a difference bit D and a borrow out Br bit. This is called full subtraction. The Boolean functions describing the full-subtracter are:

$D = (x \oplus y) \oplus Cin$ $\qquad$ $Br = A'B + A'(Cin) + B(Cin)$

I. TO REALIZE HALF ADDER

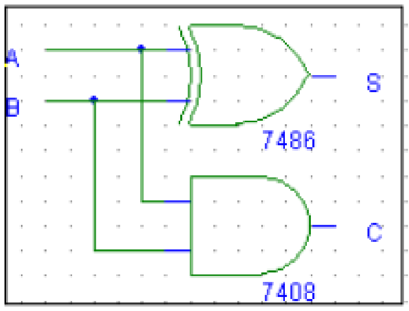
TRUTH TABLE

INPUTS		OUTPUTS	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

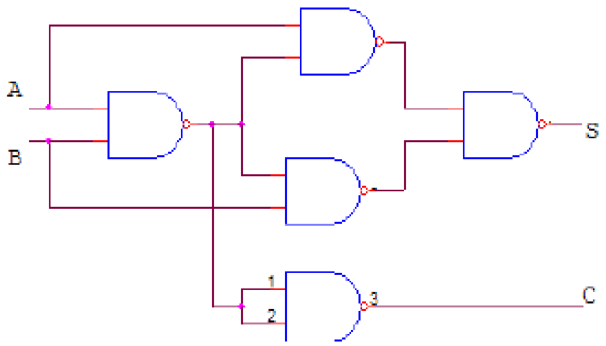
BOOLEAN EXPRESSIONS:

S=A ⊕ B
C=A B

i) Basic Gates



ii) NAND Gates



II. FULL ADDER

TRUTH TABLE

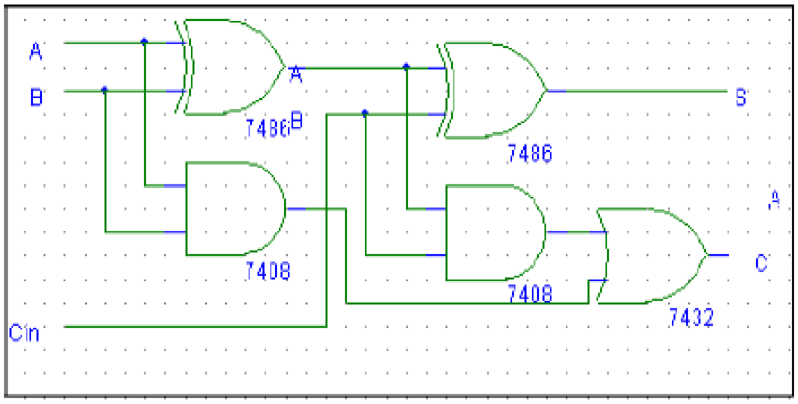
INPUTS			OUTPUTS	
A	B	Cin	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

BOOLEAN EXPRESSIONS:

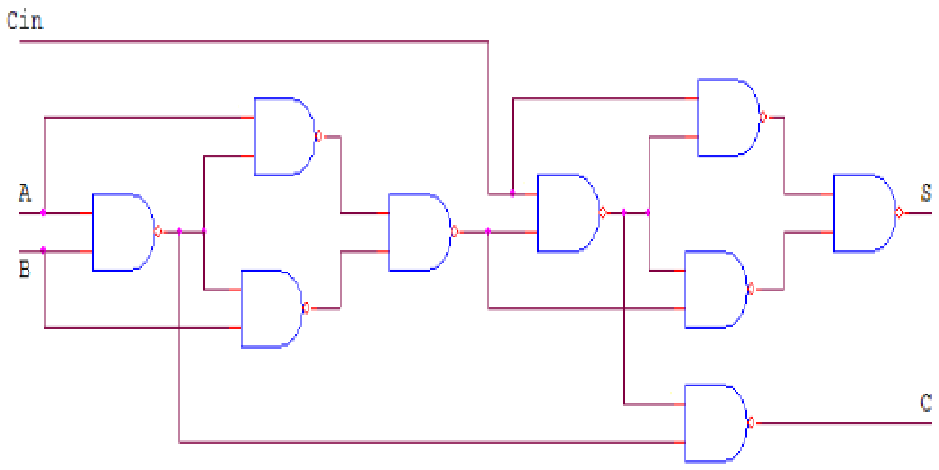
S= A ⊕ B ⊕ C

C=A B + B Cin + A Cin

i)BASIC GATES



ii) NAND GATES



III. HALF SUBTRACTOR

TRUTH TABLE

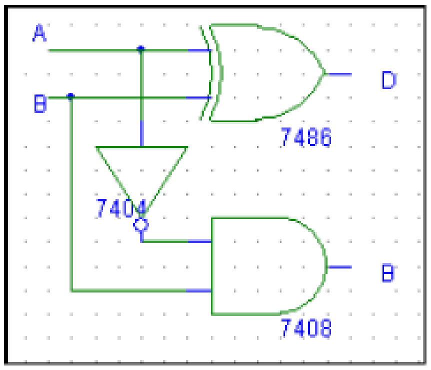
INPUTS		OUTPUTS	
A	B	D	Br
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

BOOLEAN EXPRESSIONS:

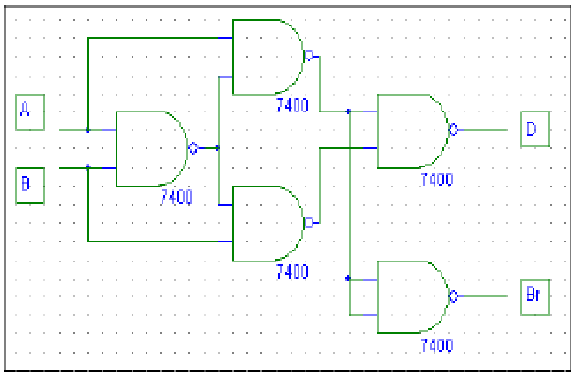
$$D = A \oplus B$$

$$Br = \bar{A}B$$

i)BASIC GATES



ii) NAND Gates



IV. FULL SUBTRACTOR

TRUTH TABLE

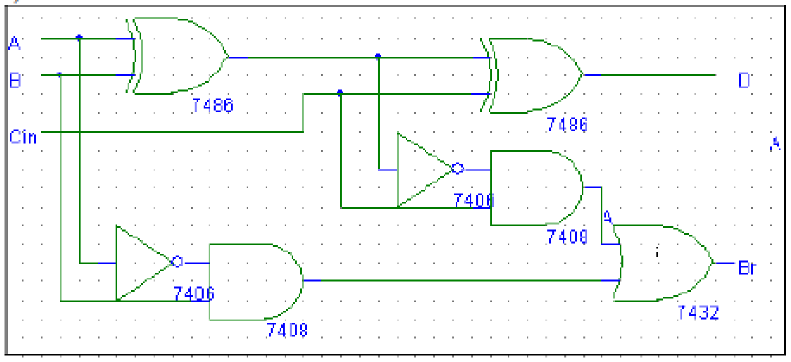
INPUTS			OUTPUTS	
A	B	Cin	D	Br
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

BOOLEAN EXPRESSIONS:

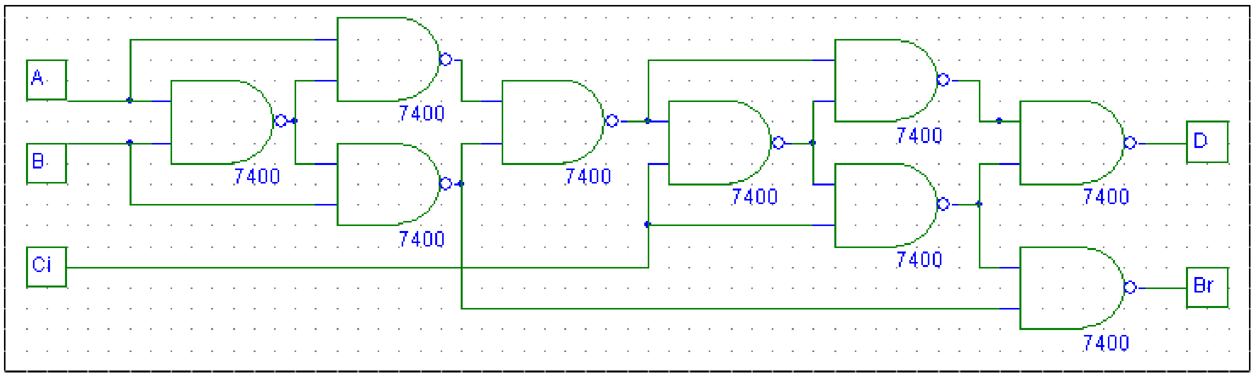
$$D = A \oplus B \oplus C$$

$$Br = \bar{A}\, B + B\, Cin + \bar{A}\, Cin$$

i) BASIC GATES



ii) To Realize the Full subtractor using NAND Gates only



PROCEDURE:

- Check the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

RESULT:

QUESTIONS:

1) What is a half adder?
2) What is a full adder?
3) What are the applications of adders?
4) What is a half subtractor?
5) What is a full subtractor?
6) What are the applications of subtractors?
7) Obtain the minimal expression for above circuits.
8) Realize a full adder using two half adders
9) Realize a full subtractors using two half subtractors

EXPERIMENT: 5 **BINARY TO GREY CONVERTER**

AIM: To realize Binary to Gray code converter and vice versa.

<u>LEARNING OBJECTIVE:</u>

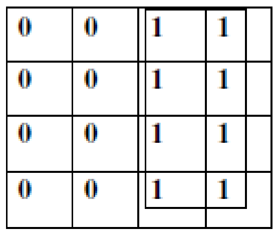
To learn the importance of non-weightedcode

To learn to generate gray code

<u>COMPONENTS REQUIRED:</u>

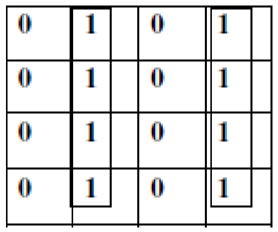
IC 7400, IC 7486, and IC 7408, Patch Cords & IC Trainer Kit

I) BINARY TO GRAY CONVERSION



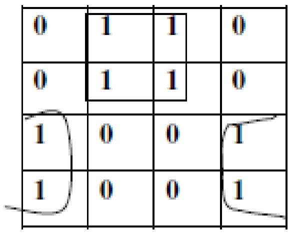
0	0	1	1
0	0	1	1
0	0	1	1
0	0	1	1

G3 = B3



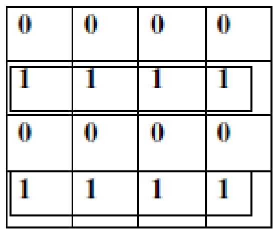
0	1	0	1
0	1	0	1
0	1	0	1
0	1	0	1

G2= B3 ⊕ B2



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

G1=B1 ⊕ B2



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

G0=B1 ⊕ B0

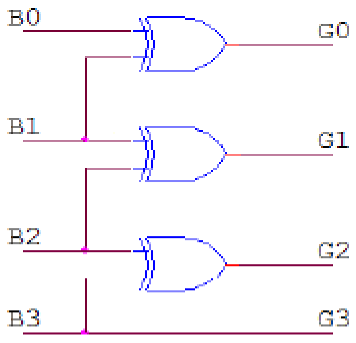
Binary				Gray			
B3	**B2**	**B1**	**B0**	**G3**	**G2**	**G1**	**G0**
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0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

BOOLEAN EXPRESSIONS:

G3=B3

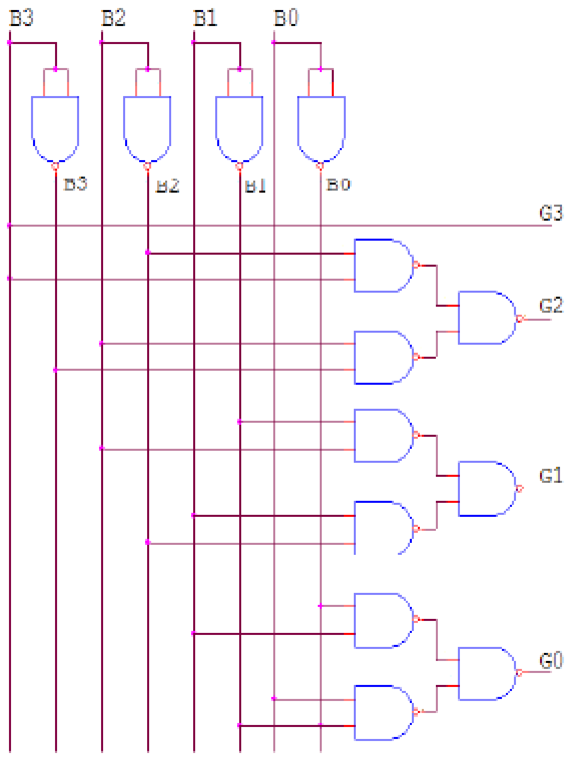
G2=B3 ⊕ B2

G1=B1 ⊕ B2; G0=B1 ⊕ B0



BINARY TO GRAY CODE USING EX-OR GATES

REALIZATION USING NAND GATES:



II) GRAY TO BINARY CONVERSION

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

B3 = G3

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

B2=G3⊕G2

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

B1=G3⊕G2⊕G1

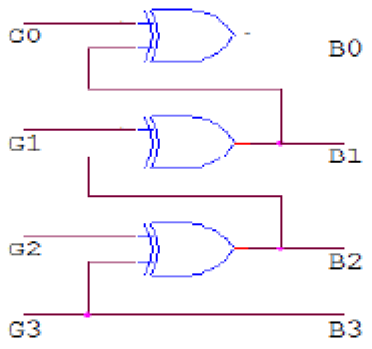
0	1	0	1
1	0	1	0
0	1	0	1
1	0	1	0

B0=G3 ⊕ G2⊕ G1⊕ G0

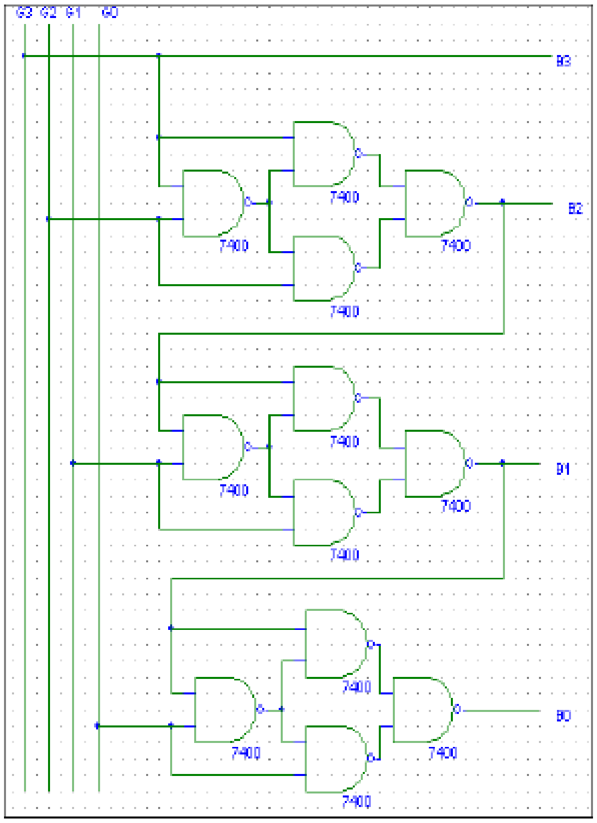
BOOLEAN EXPRESSIONS:
B3=G3
B2=G3 ⊕ G2
B1=G3 ⊕ G2⊕ G1
B0=G3 ⊕ G2⊕ G1⊕ G0

Gray				**Binary**			
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

GRAY TO BINARY CODE CONVERSION USING EX-OR GATES



REALIZATION USING NAND GATES:



PROCEDURE:

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

RESULT:

QUESTIONS:

1) What are code converters?
2) What is the necessity of code conversions?
3) What is gray code?
4) Realize the Boolean expressions for
a) Binary to gray code conversion
b) Gray to binary code conversion

EXPERIMENT: 6 MULTIPLEXER AND DEMULTIPLEXER

AIM: To design and set up the following circuit

1) To design and set up a 4:1 Multiplexer (MUX) using only NAND gates.
2) To design and set up a 1:4 Demultiplexer(DE-MUX) using only NAND gates.

LEARNING OBJECTIVE:

3)
1. To learn about various applications of multiplexer and de-multiplexer

THEORY:

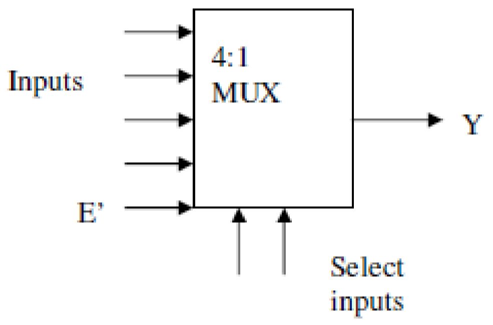
Multiplexers are very useful components in digital systems. They transfer a large number of information units over a smaller number of channels, (usually one channel) under the control of selection signals. Multiplexer means many to one. A multiplexer is a circuit with many inputs but only one output. By using control signals (select lines) we can select any input to the output. Multiplexer is also called as data selector because the output bit depends on the input data bit that is selected. The general multiplexer circuit has 2^n input signals, n control/select signals and 1 output signal.

De-multiplexers perform the opposite function of multiplexers. They transfer a small number of information units (usually one unit) over a larger number of channels under the control of selection signals. The general de-multiplexer circuit has 1 input signal, n control/select signals and 2^n output signals. De-multiplexer circuit can also be realized using a decoder circuit with enable.

COMPONENTS REQUIRED:

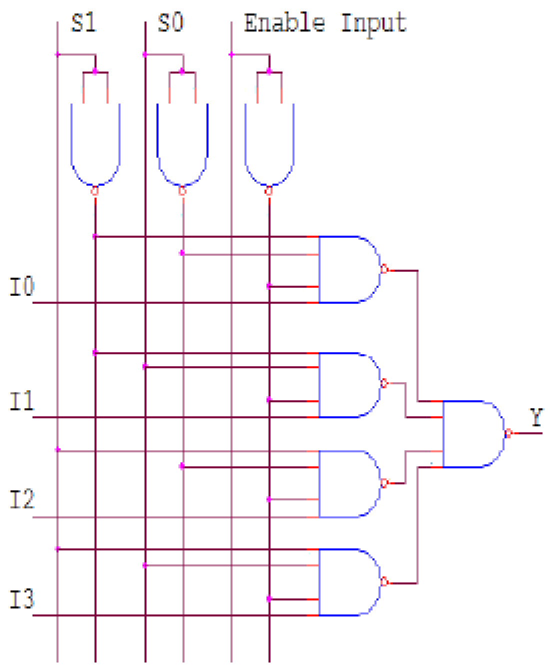
IC 7400, IC 7410, IC 7420, IC 7404, IC 74153, IC 74139, Patch Cords & IC Trainer Kit.

i) 4:1 MULTIPLEXER



Output Y= E'S1'S0'I0 + E'S1'S0I1 + E'S1S0'I2 + E'S1S0I3

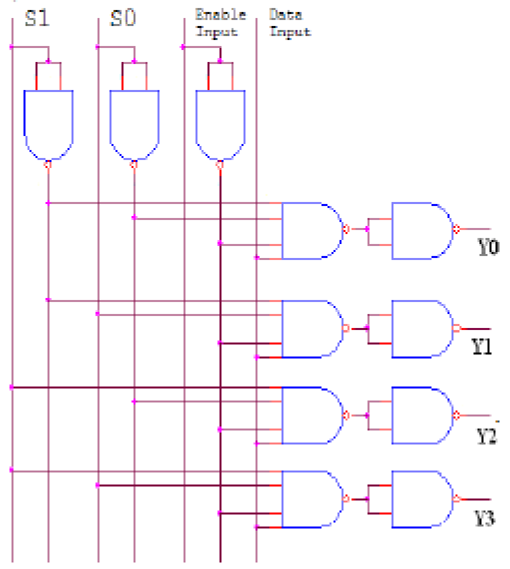
REALIZATION USING NAND GATES



TRUTH TABLE

Select Inputs		Enable Input	Inputs				Out puts
S_1	S_0	E	I_0	I_1	I_2	I_3	Y
X	X	1	X	X	X	X	0
0	0	0	0	X	X	X	0
0	0	0	1	X	X	X	1
0	1	0	X	0	X	X	0
0	1	0	X	1	X	X	1
1	0	0	X	X	0	X	0
1	0	0	X	X	1	X	1
1	1	0	X	X	X	0	0
1	1	0	X	X	X	1	1

ii) DE-MUX USING NAND GATES



Enable Inputs	Data Input	Select Inputs		Outputs			
E	D	S_1	S_0	Y_3	Y_2	Y_1	Y_0
1	0	X	X	X	X	X	X
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	1	0	0
0	1	1	1	1	0	0	0

PROCEDURE:

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

VIVA QUESTIONS:

1) What is a multiplexer?
2) What is a de-multiplexer?
3) What are the applications of multiplexer and de-multiplexer?
4) Derive the Boolean expression for multiplexer and de-multiplexer.
5) How do you realize a given function using multiplexer
6) What is the difference between multiplexer &demultiplexer?
7) In 2n to 1 multiplexer how many selection lines are there?
8) How to get higher order multiplexers?
9) Implement an 8:1 mux using 4:1 muxes?

EXPERIMENT: 7: REALIZATION OF A BOOLEAN FUNCTION USING LOGISIM

AIM:

To set up a circuit using Logisim: I(A,B,C,D) =∑(0,1,2,3,4,5,6,7,8,9,10) and show the truthtables.

OBJECTIVE:

1.To learn about various applications of Logisim

COMPONENTSREQUIRED:

Logisim Software

THEORY:

The basic logic gates are the building blocks of more complex logic circuits. These logic gates perform the basic Boolean functions, such as AND, OR, NAND, NOR, Inversion, Exclusive-OR, Exclusive-NOR. All the combinational circuits are designed using these gates.

PROCEDURE:

- Solve the Boolean algebra expression using K Map
- Using different components draw the circuit in Logisim
- Simulate the circuit to verify

RESULT:

The simulation obtained from the circuit can be verified from the truth tables.

- Attach the copy of the circuit designed in Logisim.

EXPERIMENT: 8 FLIP FLOPS

AIM: Truth Table verification of
1) RS Flip Flop
2) T type Flip Flop.
3) D type Flip Flop.
4) JK Flip Flop.

OBJECTIVE:

- To learn about various Flip-Flops
- To learn about applications of FFs
- Conversion of one type of Flip flop to another

THEORY:

Logic circuits that incorporate memory cells are called *sequential logic circuits*; their output depends not only upon the present value of the input but also upon the previous values. Sequential logic circuits often require a timing generator (a clock) for their operation. The latch (flip-flop) is a basic bi-stable memory element widely used in sequential logic circuits. Usually there are two outputs, Q and its complementary value. Some of the most widely used latches are listed below.

SR LATCH:

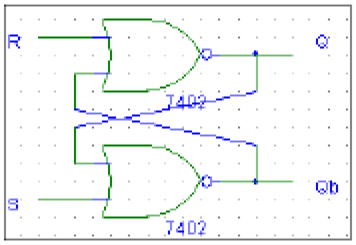
An S-R latch consists of two cross-coupled NOR gates. An S-R flip-flop can also be design using cross-coupled NAND gates as shown. The truth tables of the circuits are shown below.

A clocked S-R flip-flop has an additional clock input so that the S and R inputs are active only when the clock is high. When the clock goes low, the state of flip-flop is latched and cannot change until the clock goes high again. Therefore, the clocked S-R flip-flop is also called "enabled" S-R flip-flop.

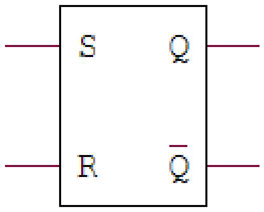
A D latch combines the S and R inputs of an S-R latch into one input by adding an inverter. When the clock is high, the output follows the D input, and when the clock goes low, the state is latched.

A S-R flip-flop can be converted to T-flip flop by connecting S input to Qb and R to Q.

1) S-R LATCH:



(A) LOGIC DIAGRAM



(B) SYMBOL

THEORY:

Logic circuits that incorporate memory cells are called ***sequential logic circuits***; their output depends not only upon the present value of the input but also upon the previous values. Sequential logic circuits often require a timing generator (a clock) for their operation. The latch (flip-flop) is a basic bi-stable memory element widely used in sequential logic circuits. Usually there are two outputs, Q and its complementary value. Some of the most widely used latches are listed below.

SR LATCH:

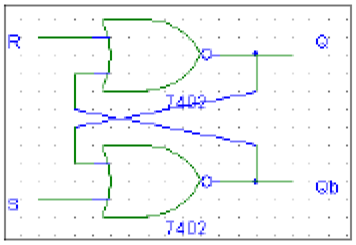
An S-R latch consists of two cross-coupled NOR gates. An S-R flip-flop can also be design using cross-coupled NAND gates as shown. The truth tables of the circuits are shown below.

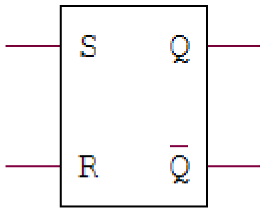
A clocked S-R flip-flop has an additional clock input so that the S and R inputs are active only when the clock is high. When the clock goes low, the state of flip-flop is latched and cannot change until the clock goes high again. Therefore, the clocked S-R flip-flop is also called "enabled" S-R flip-flop.

A D latch combines the S and R inputs of an S-R latch into one input by adding an inverter. When the clock is high, the output follows the D input, and when the clock goes low, the state is latched.

A S-R flip-flop can be converted to T-flip flop by connecting S input to Qb and R to Q.

1) S-R LATCH:



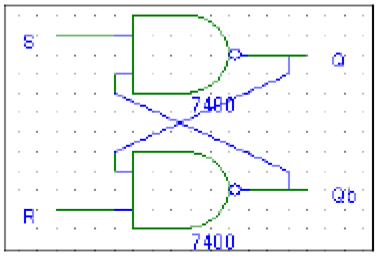


(A) LOGIC DIAGRAM **(B) SYMBOL**

TRUTH TABLE

S	R	Q+	$\overline{Q}$b+
0	0	Q	$\overline{Q}$b
0	1	0	1
1	0	1	0
1	1	0*	0*

$\overline{S}\overline{R}$ LATCH:

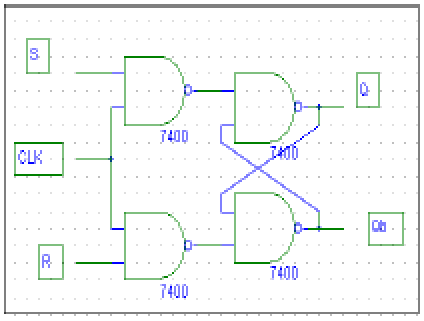


TRUTH TABLE

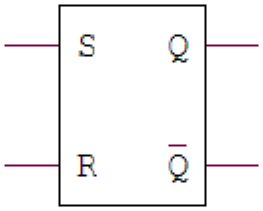
S	R	Q+	$\overline{Q}$b+
0	0	1*	1*
0	1	1	0
1	0	0	1
1	1	Q	$\overline{Q}$b

2) SR FLIP FLOP:

CIRCUIT DIAGRAM:



(A) LOGIC DIAGRAM



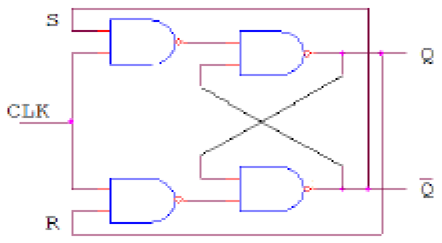
(B) SYMBOL

TRUTH TABLE

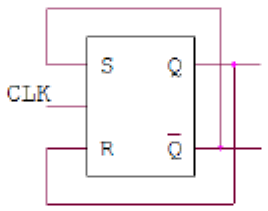
S	R	Q+	$\overline{Q}$b+
0	0	Q	$\overline{Q}$b
0	1	0	1
1	0	1	0
1	1	0*	0*

3) CONVERSION OF SR-FLIP FLOP TO T-FLIP FLOP (Toggle)

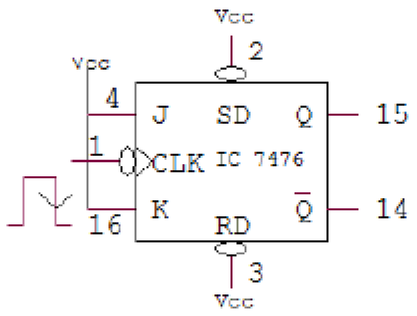
LOGIC DIAGRAM



SYMBOL



T FLIP FLOP USING IC 7476

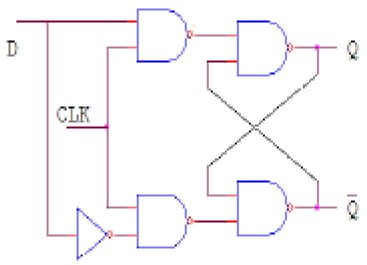


TRUTH TABLE

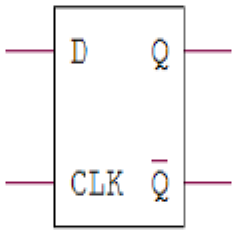
T	Qn + 1
0	Qn
1	$\overline{Qn}$

4) CONVERSION OF SR-FLIP FLOP TO D-FLIP FLOP :

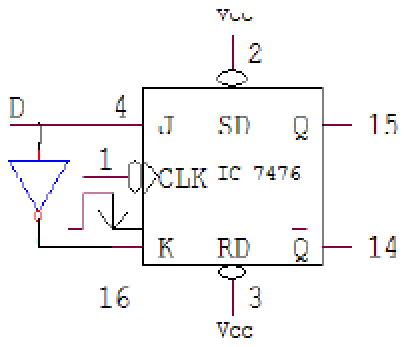
LOGIC DIAGRAM



SYMBOL



D FLIP FLOP USING IC 7476

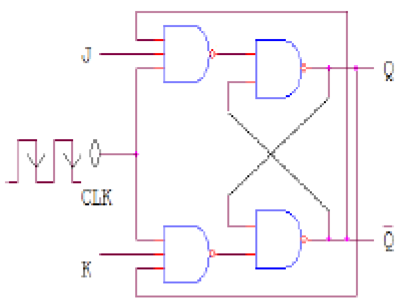


TRUTH TABLE

CLOCK	D	Q+	$\overline{Q+}$
0	X	Q	$\overline{Q}$
1	0	0	1
1	1	1	0

5. CONVERSION OF SR-FLIP FLOP TO JK-FLIP FLOP

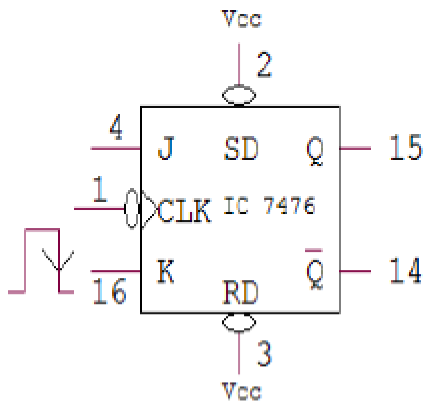
LOGIC DIAGRAM



TRUTH TABLE

Clock	J	K	Q+	Q'+	Comment
1	0	0	Q	Q'	No Change
1	0	1	0	1	Reset
1	1	0	1	0	Set
1	1	1	Q'	Q	Toggle

LOGIC DIAGRAM



TRUTH TABLE

SD	RD	Clock	J	K	Q	Q'	Comment
0	0	Not Allowed					
0	1	X	X	X	1	0	Set
1	0	X	X	X	0	1	Reset
1	1	1	0	0	NC	NC	Memory
1	1	1	0	1	0	1	Reset
1	1	1	1	0	1	0	Set
1	1	1	1	1	Q'	Q	Toggle

PROCEDURE:

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

VIVA QUESTIONS:

1. What is the difference between Flip-Flop & latch?
2. Give examples for synchronous & asynchronous inputs?
3. What are the applications of different Flip-Flops?
4. What is the advantage of Edge triggering over level triggering?
5. What is the relation between propagation delay & clock frequency of flip-flop?
6. What is race around in flip-flop & how to over come it?
7. Convert the J K Flip-Flop into D flip-flop and T flip-flop?
8. List the functions of asynchronous inputs?