Indira Gandhi Delhi Technical University for Women

(Established by Govt. of Delhi vide Act 09 of 2012)

(Formerly Indira Gandhi Institute of Technology)

Kashmere Gate, Delhi-110006

LABORATORY FILE

For

Computer Organization & Architecture

BCS 254

SUBMITTED TO: -	**SUBMITTED BY:-**
Ms. BHAWNA NARWAL	**OSHIN SAINI**
IT DEPARTMENT	**03001032018**

INDEX

Experiment No. 1

Aim: To verify the truth tables of AND, OR, NAND, NOR, XOR, XNOR, NOT and BUFFER GATES.

Apparatus required: Trainers kit with required integrated circuits and connecting wires.

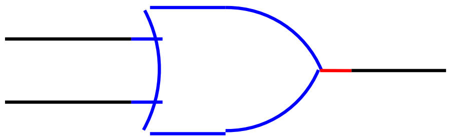
Theory:

A logic gate is a digital circuit that follows certain logical relationship between one or more than one input and the output. The input and the output are voltages which can be only on one of the possible two states. Thus the input or output may be low (called logic 0) or high (called logic 1). Some of the commonly used gates are:

OR GATE (2 INPUTS)

It is a logic gate based on logical OR operation. It is a logic gate in which either of the two inputs (A or B) is high (logic one) to get high (logic one) output.

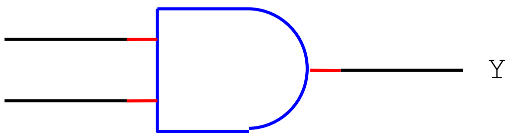
Y = A + B



AND GATE (2 INPUTS)

It is a logic gate based on logical and operation. It is a logic gate which gives high (logic 1) output when both inputs (A and B) are high.

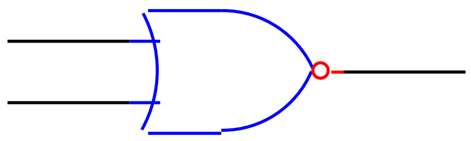
Y = A. B



NOR GATE (2 INPUTS)

It is a logic based on neither logical nor operation. It is a logic gate which gives high output only when both inputs are low.

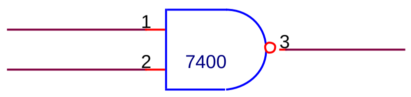
Y = (A + B)’



NAND GATE (2 INPUT)

It is a logic based on logical Nand operation. It is a logic gate which gives low output only when both inputs are high.

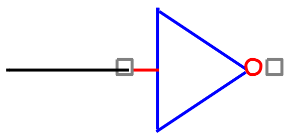
Y = (A. B)’



NOT GATE

It is a logic based on logical not operation. It is a logic gate which gives output (Y) the complement of the input (A). It is also known as inverter.

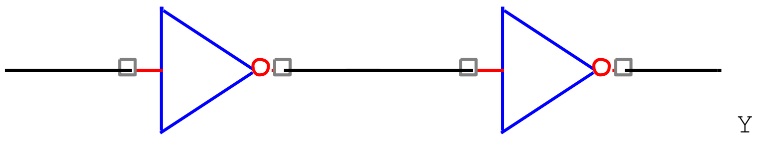
Y = A’



BUFFER GATE

It is a logic gate consisting of two NOT gates. The output of first NOT gate is fed as input into the next NOT gate. This logic gate yields the output same as the input. This is used to increase the current handling capacity and provide delay while designing.

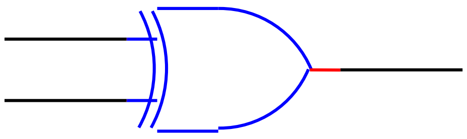
Y = (A')' = A



XOR GATE (2 INPUT)

It is a logic gate based on exclusive or operation. It is a logic gate which yields a high output when inputs are different and low output when both the inputs are same.

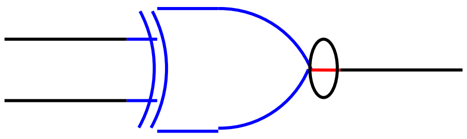
Y = A'B + AB'



X-NOR GATE

It is a logic gate based on exclusive nor gate. It is a logic gate which yields a high output when both inputs are same and low output when both inputs are different.

Y = (A'B + AB')'



Procedure:

1. Make connections on trainer kit for a required gate say AND gate.

2. Connect the terminals 2 and 3 of the AND gate IC on trainer kit to input terminals (1 and 2 Pin) and output to pin no. 3. (i.e. the required LEDs to required pin nos.)

3. Connect the pin no.7 to ground and pin no.14 to the supply (Vcc). Switch on the trainer kit.

4. Observe the output of the gate for various sets of inputs. Note the observations.

5. Repeat all the above steps or all other gates using the appropriate IC and note the observation.

Observations:

(1) AND GATE

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

(2) OR GATE

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

(3) NAND GATE

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

(4) NOR GATE

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

(5) XOR GATE

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

(6) XNOR GATE

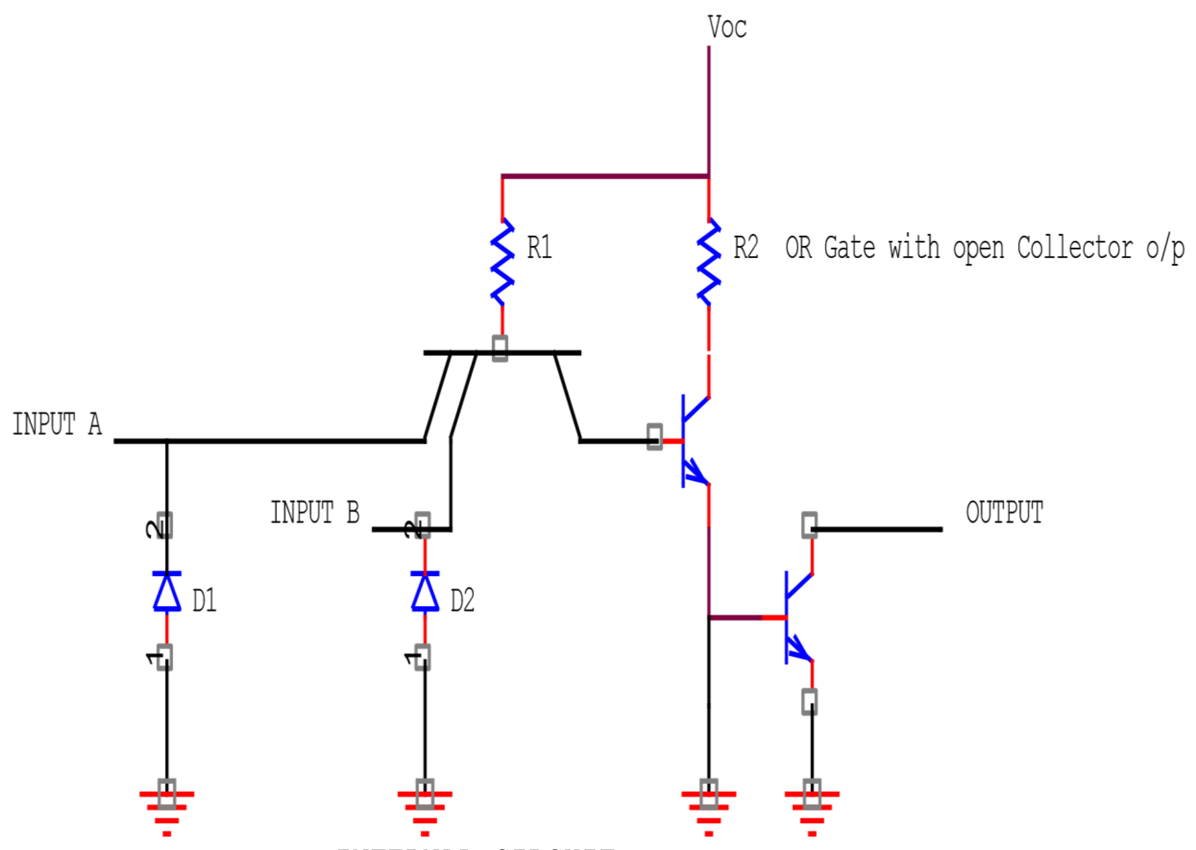
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

(7) NOT GATE

A	Y
0	1
1	0

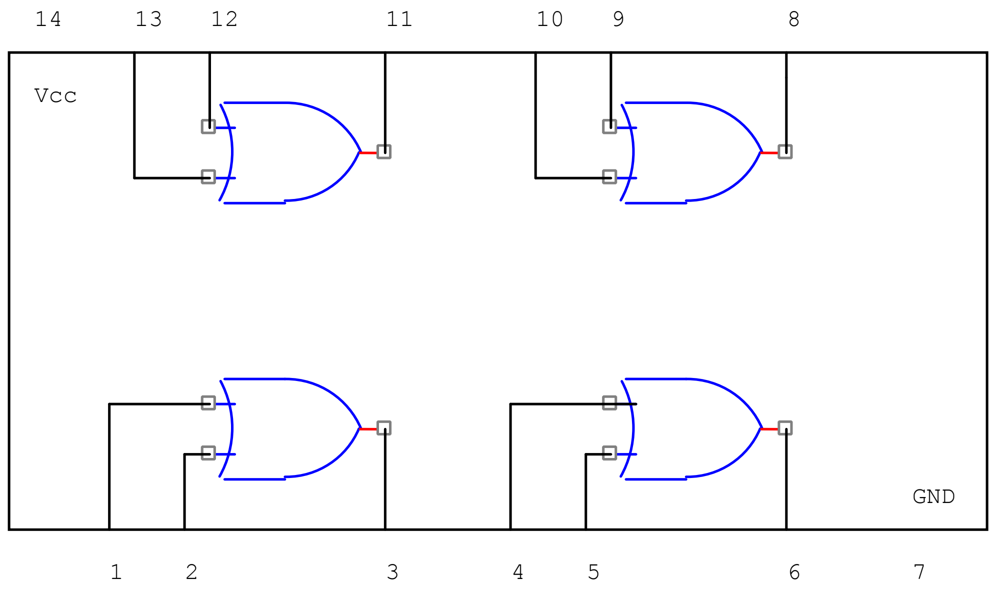
(8) BUFFER GATE

A	Y
0	0
1	1



INTERNAL CIRCUIT

14 PIN DIAGRAM (7432)



Result: Truth tables were verified for AND, OR, NAND, NOR, XOR, XNOR, NOT and BUFFER logic gates.

PRECAUTIONS:

1. All the IC’s should be handled carefully.
2. All the connection should be tight.
3. Supply should be given after all connections are made.
4. Use IC plucker when remove the IC from Bread Board.

Experiment No. 2

Aim: To verify various laws of Boolean algebra.

Apparatus required: Trainers, required IC's (7432, 7408 and 7404) and Connecting wires.

Theory: The various laws of Boolean algebra are as followed:

1. **COMMUTATIVE LAW**

   If A and B are two inputs then
   $A + B = B + A$
   $AB = BA$

2. **ASSOCIATIVE LAW**

   If A, B and C are input variables then
   $A + (B + C) = (A + B) + C$
   $A (BC) = (AB) C$

3. **DISTRIBUTIVE LAW**

   If A, B and C are input variables then
   $A + (BC) = (A + B) (A + C)$
   $A (B + C) = AB + AC$

4. **OR LAW**

   According to this law
   (I) $A + 0 = A$ (II) $A + A = A$
   (III) $A + 1 = 1$ (IV) $A + A' = 1$

5. **AND LAW**

   According to this law
   (I) $A 1 = A$ (II) $A. A = A$
   (III) $A. 0 = 0$ (IV) $A. A' = 0$

6. **<u>DEMORGANS LAW</u>**

According to this law for A and B as two input variables

(A + B)’ = A’. B’

(AB)’ = A’ + B’

<u>Procedure:</u>

1. Make the connections required for the given law with the help of required ICs say for commutative law.
2. Connect the terminal 1 and 2 of the OR gate IC on trainers kit to input terminal and output to pin 3 (i.e. required LED to required pin).
3. Make other connections for ground (pin 7) and supply (pin 14) and switch on the kit.
4. Observe the output for various sets of inputs. Note them down. Now exchange the input terminals and compare the output with the earlier observations and verify.
5. Repeat the steps required for verifying the various laws and note the observations.

Observations:

1. **COMMUTATIVE LAW**

(i) A + B = B + A

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

(ii) AB = BA

A	B	AB	BA
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

2) **ASSOCIATIVE LAW**

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

A	B	C	A + (B + C)	(A + B) + C
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

(ii) A (BC) = (AB) C

A	B	C	A(BC)	(AB)C
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0

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

2. DISTRIBUTIVE LAW

(i) A + (BC) = (A + B) (A + C)

A	B	C	A + (BC)	(A + B)(A + C)
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

(ii) A (B + C) = AB + AC

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

3. OR LAW

(i) A + 0 = A

A	0	A + 0
0	0	0
1	0	1

(ii) A + A = A

A	A	A + A
0	0	0
1	1	1

(iii) A + 1 = 1

A	1	A + 1
0	1	1
1	1	1

(IV) A + A’ = 1

A	A’	A + A’
0	1	1
1	0	1

4. AND LAW

(i) A. 1 = A

A	1	A. 1
0	1	0
1	1	1

(ii) A. A = A

A	A	A. A
0	0	0
1	1	1

(iii) A. 0 = 0

A	0	A. 0
0	0	0
1	0	0

(iv) A. A’ = 0

A	A’	A. A’
0	1	0
1	0	0

5. DEMORGANS LAW

(i) (A + B)’ = A’. B’

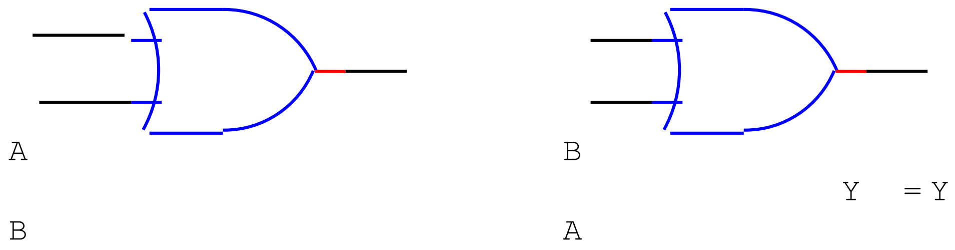
A	B	(A + B)’	A’. B’
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

(ii) (AB)’ = A’ + B’

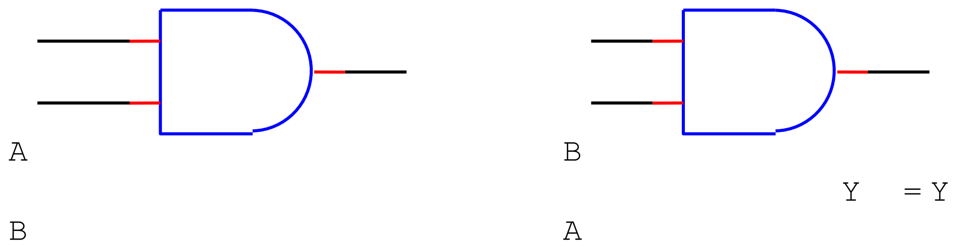
A	B	(AB)’	A’ + B’
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

COMMUTATIVE LAW VERIFICATION

(i) A + B = B + A



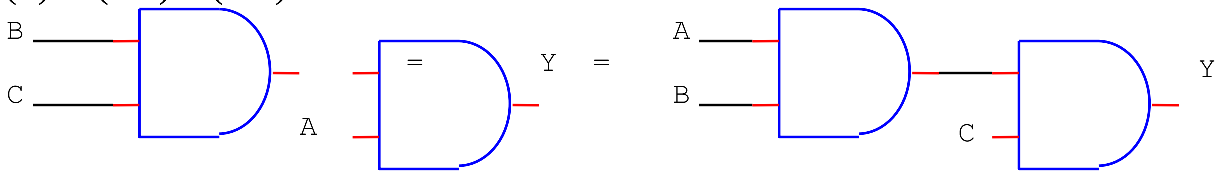
(ii) AB = BA



ASSOCIATIVE LAW VERIFICATION

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

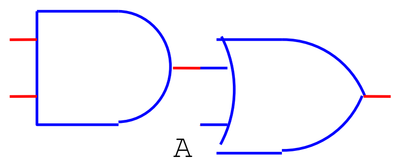
(ii) A (BC) = (AB) C



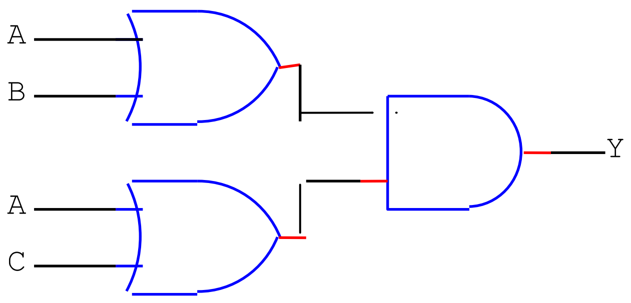
DISTRIBUTIVE LAW VERIFICATION

(i) A + (BC) = (A + B) (A + C)

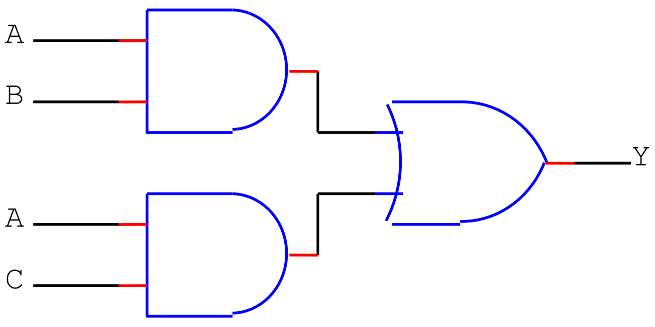
B



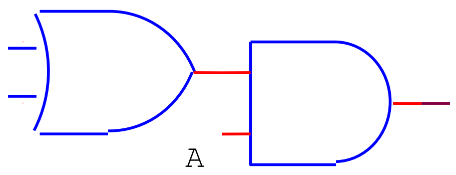
C Y =



=



(ii) A (B + C) = AB + AC

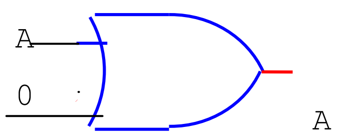


B

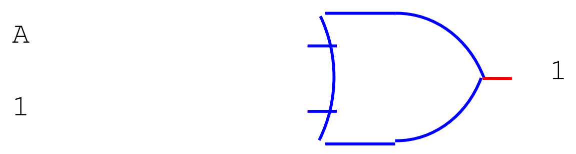
C Y

OR LAW

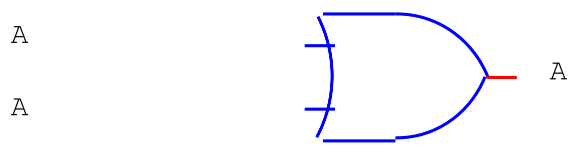
(i) A + 0 = A



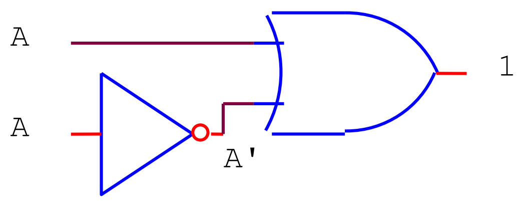
(ii) A + 1



(iii) A + A = A

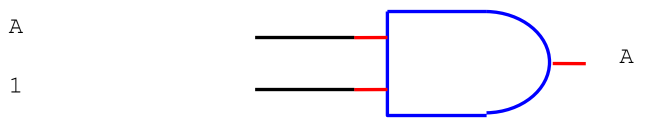


(iv) A + A’ = 1

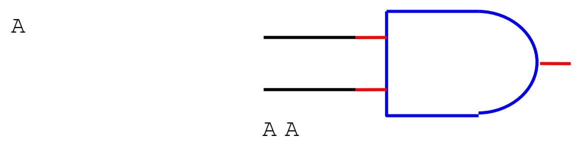


AND LAW

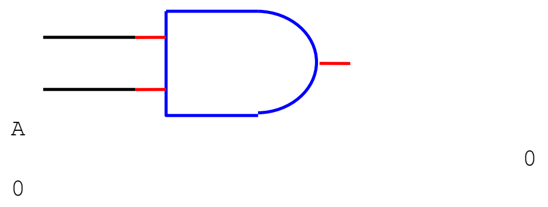
(i) A.1 = A



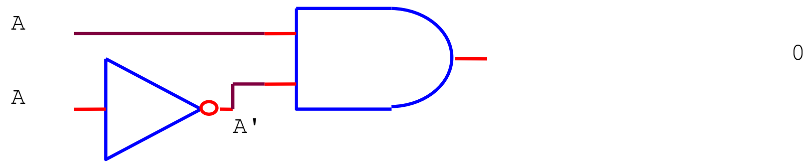
(ii) A.A = A



(iii) A.0 = 0

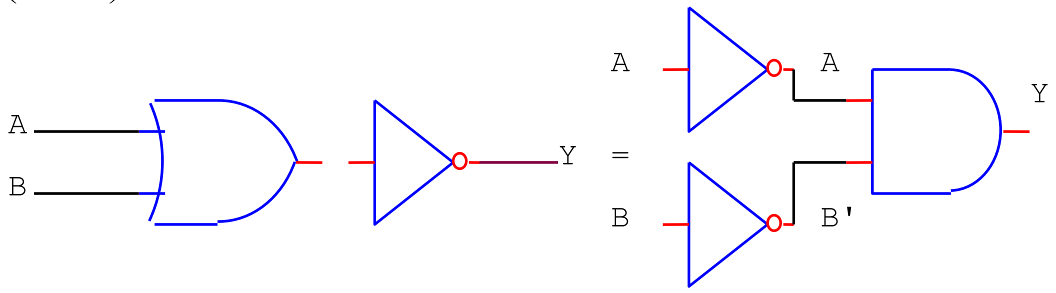


(iv) A.A’ = 0

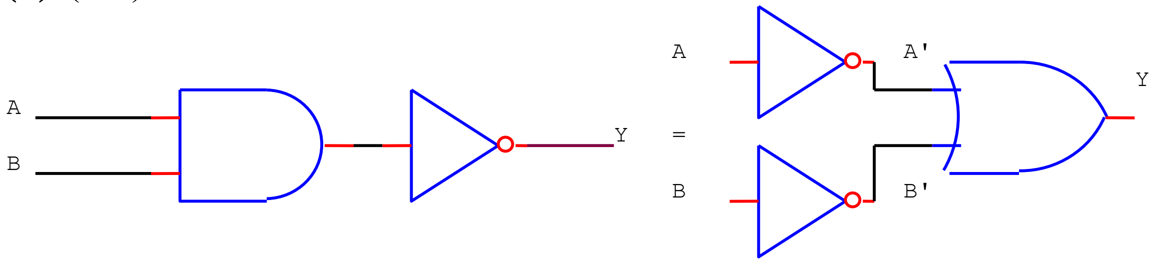


DEMORGANS LAW

(i) (A + B)’ = A’. B’



(ii) (AB)’ = A’ + B’



Result: The laws of Boolean algebra are verified.

PRECAUTIONS:

1. All the IC’s should be handled carefully.
2. All the connection should be tight.
3. Supply should be given after all connections are made.
4. Use IC plucker when remove the IC from Bread Board.

Experiment No. 3

Aim: To design AND, OR, NOT, XOR, XNOR, NAND (NOR) gates using NOR and NAND gates.

Apparatus required: IC (No. 7400 and 7402), connecting wires, and bread board.

Theory:

It is possible to design all the logic gates using NAND and NOR gates. So these two gates are called universal gates.

USING NAND GATE

1. **OR GATE**

   OR gate can be designed using 3 NAND gates. For OR gate with A and B as input and Y be the output then

   Y = A + B
   Y = ((A+B)')' = (A'.B')'   (using Demorgans Law)
   = (A NAND A) NAND (B NAND B)

2. **AND GATE**

   AND gate can be designed using 2 NAND gates.
   Here Y = A.B
   Y = ((A.B)')' = (A NAND B) NAND (A NAND B)

3. **NOT GATE**

   NOT gate can be designed using only 1 NAND gate. For NOT gate:
   Y = A'
   Y = ((A)')' = (A.A)'
   Y = A NAND A

4. **NOR GATE**

   NOR gate can be designed using 4 NAND gates. For NOR gate:
   Y = (A+B)'

Y = [(A NAND A) NAND (B NAND B)] NAND [(A NAND A) NAND (B NAND B)]

5. **XOR GATE**

XOR gate can be designed using 4 NAND gates. For XOR gates:
Y = A’B + AB’
Y = [(A NAND (A NAND B)) NAND (B NAND (A NAND B))]

6. **XNOR GATE**

XNOR gate can be designed using 5 NAND gates. Here
Y = [(A NAND (A NAND B)) NAND (B NAND (A NAND B))] NAND [(A NAND (A NAND B)) NAND (B NAND (A NAND B))]

USING NOR GATE

1. **OR GATE**

OR gate can be designed using 2 NOR gates. Here
Y = (A NOR B) NOR (A NOR B)

2. **AND GATE**

AND gate can be designed using 3 NOR gates. Here
Y = (A NOR A) NOR (B NOR B)

3. **NOT GATE**

NOT gate can be designed using only 1 NOR gate. For NOT gate: Y = A NOR A

4. **NAND GATE**

NAND gate can be designed using 4 NOR gates.
Y = [(A NOR A) NOR (B NOR B)] NOR [(A NOR A) NOR (B NOR B)]

5. **XOR GATE**

XOR gate can be designed using 5 NOR gates. For XOR gates:
Y = [(A NOR (A NOR B)) NOR (B NOR (A NOR B))] NOR
[(A NOR (A NOR B)) NOR (B NOR (A NOR B))]

6. XNOR GATE

XNOR gate can be designed using 4 NOR gates. Here
Y = = [(A NOR (A NOR B)) NOR (B NOR (A NOR B))]

Procedure:

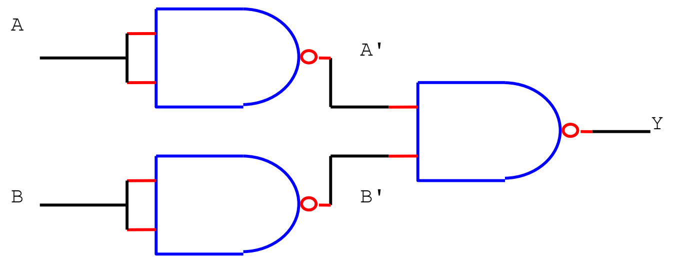
1. For designing the required gate from NAND or NOR gate, connect the required IC.
2. Connect the gate in the required manner with the help of connecting wires and make proper connections for ground (pin 7) and supply (pin 14).
3. Connect the inputs and outputs to the required LEDs. Switch on the supply.
4. Note the output for various combinations of input, note the observations and compare them with the observations of original gate.
5. Repeat the above steps for all other gates and note the observations.

Observations:

DESIGN OF GATES USING NAND GATE

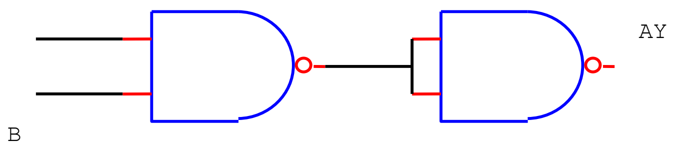
(1) **OR GATE**

Y = A + B



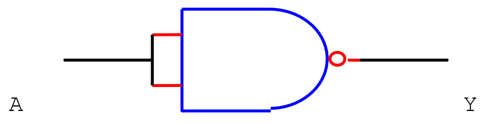
(2) **AND GATE**

Y = A. B .



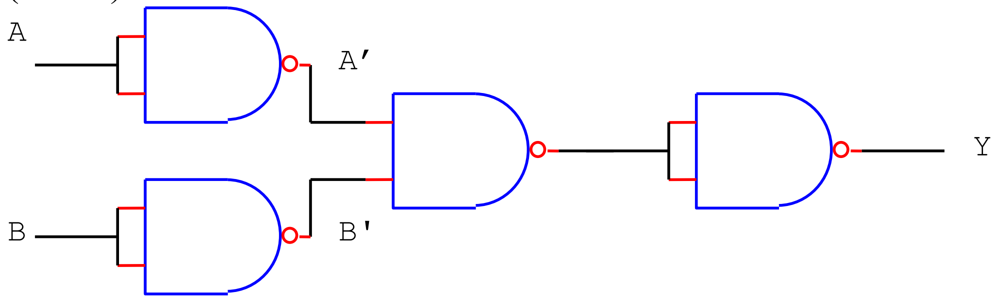
(3) **NOT GATE**

Y = A'



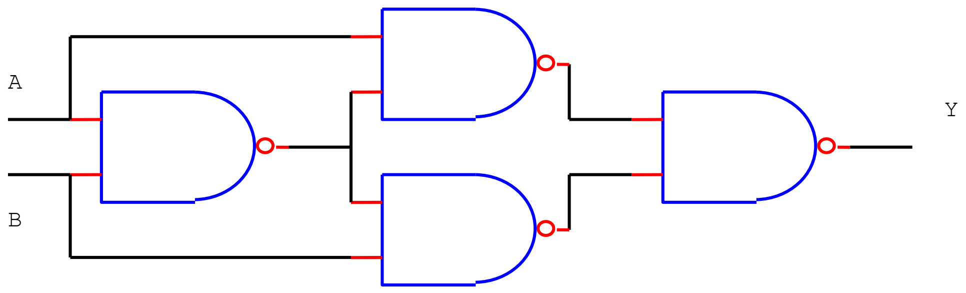
(4) **NOR GATE**

Y = (A + B)'



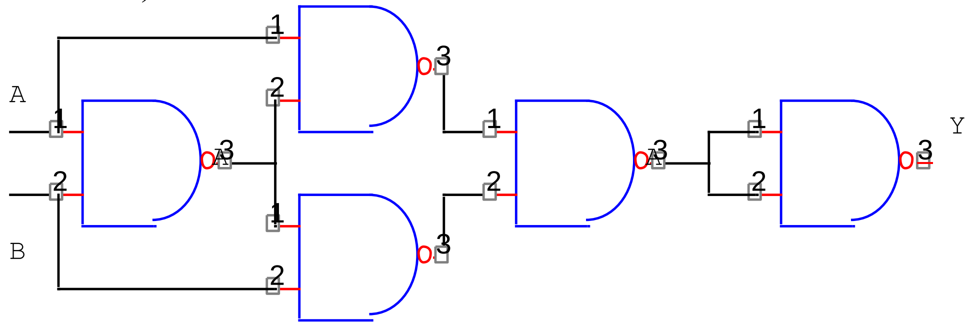
(5) **XOR GATE**

Y = A'B + AB'



(6) **XNOR GATE**

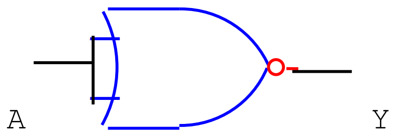
Y = (A’B + AB’)’



DESIGN OF BASIC GATES USING NOR GATE

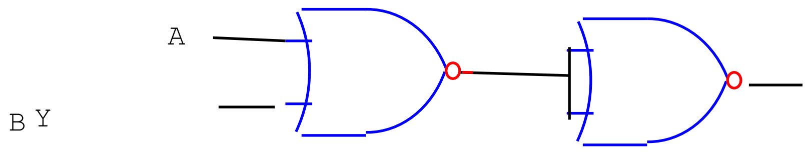
(1) **NOT GATE**

Y = A’



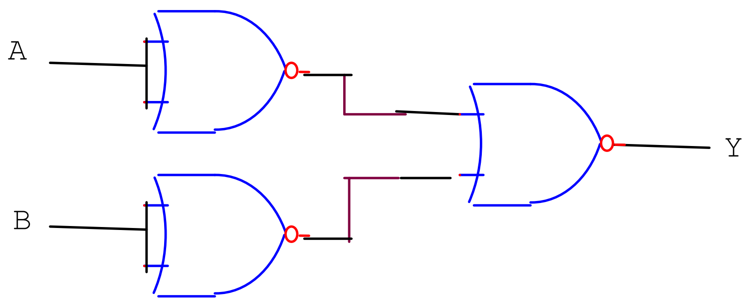
(2) **OR GATE**

Y = A + B



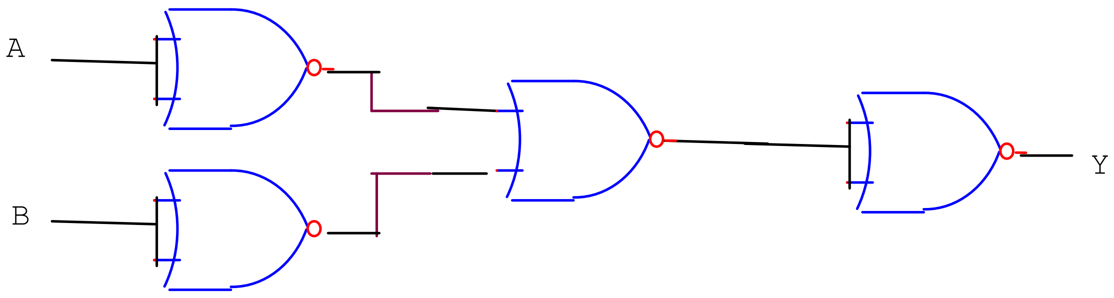
(3) **AND GATE**

Y = A.B



(4) **NAND GATE**

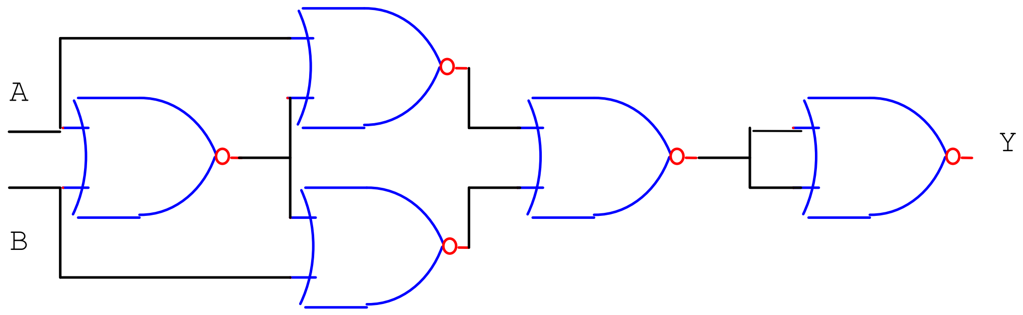
Y = (A.B)’



(5) **XOR GATE**

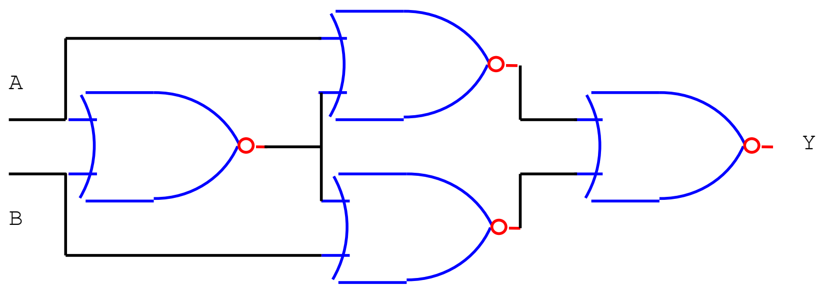
Y = A’B + AB’

-



(6) **XNOR GATE**

Y = (A’B + AB’)’



Result: All the gates (AND, OR, NOT, XOR, XNOR, NAND (NOR)) were designed using NAND and NOR gates and their truth tables verified.

PRECAUTIONS:

1. All the IC's should be handled carefully.
2. All the connection should be tight.
3. Supply should be given after all connections are made.
4. Use IC plucker when remove the IC from Bread Board.

Experiment No. 4

Aim: To design a Half adder circuit and verify its truth table. Design a full adder circuit using half adders.

Apparatus required: IC (No. 7432, 7408, 7404 and 7486), connecting wires, and trainers kit.

Theory:

HALF ADDER

A half adder is a combinational circuit that programs the addition of two bits. The circuit has two binary inputs and two binary outputs. The input variables (A and B) designate the augend and addend bits and the output variables (S and C) are sum and carry respectively. The C output is 1 only when both inputs are 1 and S represents the last significant bit of the sum.

The simplified Boolean functions for two outputs as obtained from the truth table are:

$S = A'B + AB'$

$C = AB$

FULL ADDER

A full adder is a combinational circuit that performs the addition of 3 bits. The circuit has 3 input variables (A, B and C) and two output variables (S as sum and C as carry). The third input C represents the carry from previous lower significant position. The Boolean functions for two outputs obtained from truth table are:

$S = A' (B'C + BC') + A (B'C' + BC)$

S = A XOR B XOR C

and

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

$= AB (C + C') BC (A + A') +AC (B + B')$

$= AB + BC + AC$

It can be implemented using two half adders and one OR gate. The output of first adder is sent as input to next half adder to obtain final sum.

Procedure:

1. For designing half adder, connect the required IC.
2. Connect the gates in the required manner with the help of connecting wires and make proper connections for ground (pin 7) and supply (pin 14).
3. Connect the input and output to required LEDs. Switch on the supply.
4. Note the output for various combinations of input variables.
5. Repeat the steps for full adder and note down the observations.

Observations:

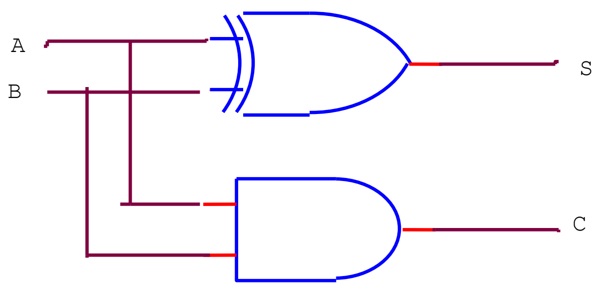
Truth table of Half Adder

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Truth table of Full Adder

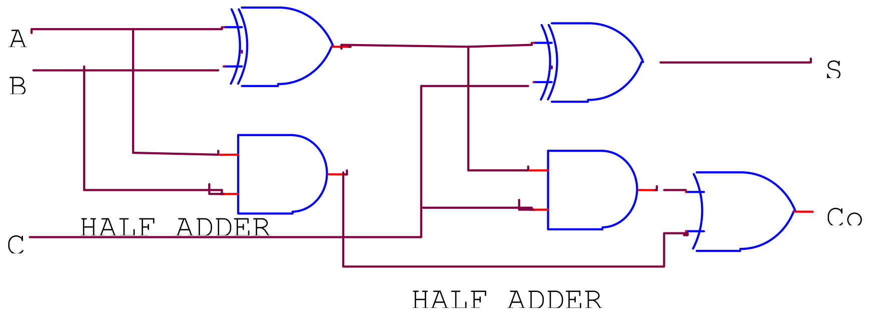
A	B	C	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

HALF ADDER



FULL ADDER

FULL ADDER USING TWO HALF ADDERS



Result: A half adder was designed and its truth table was verified. A full adder was also designed using half adders and its truth table is verified.

PRECAUTIONS:

1. All the IC's should be handled carefully.
2. All the connection should be tight.
3. Supply should be given after all connections are made.
4. Use IC plucker when remove the IC from Bread Board.

CONCLUSION:

Using a half Adder we can realize the addition of two single bit numbers. Using Full Adder circuit, we can realize the addition of n single bit numbers.

Experiment No. 5

Aim: To design a Half subtractor circuit and verify its truth table. Design a full subtractor circuit using half subtractor.

Apparatus required: IC (No. 7432, 7404, 7408 and 7486), connecting wires, and trainers kit.

Theory:

HALF SUBTRACTOR

A half subtractor is a combinational circuit that programs the subtraction of two bits. The circuit has two binary inputs (A and B) and two binary outputs (D as difference and B_o as borrow).

The simplified Boolean functions for two outputs as obtained from the truth table are:

$D = A'B + AB'$

$B_o = A'B$

FULL SUBTRACTOR

A full subtractor is a combinational circuit that performs the subtraction of 3 bits. The circuit has 3 input variables (A, B and C) and two output variables (D as difference and B_o as borrow). The Boolean functions for two outputs obtained from truth table are:

$S = A' (B'C' + BC) + A' (B'C + BC')$

$S = A \text{ XOR } B \text{ XOR } C$

And

$B_o = A'B'C + A'BC' + A'BC$

$\quad = A'C + A'B + BC$

A full subtractor can also be designed using two subtractor and an OR gate.

Procedure:

1. Connect the required ICs and make connections of gates in required manner and make proper connections for ground (pin 7) and supply (pin 14).

2. Connect the input and output to required LEDs. Switch on the supply.

3. Note the output for various combinations of input variables.

4. Repeat the steps for full adder and note down the observations.

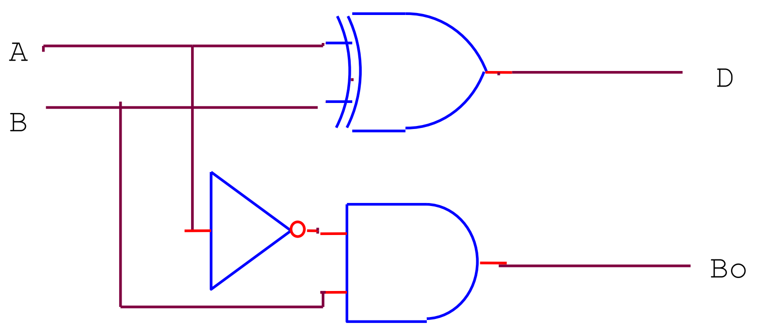
Observations:

Truth table of Half Subtractor

A	B	D	B_o
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	0

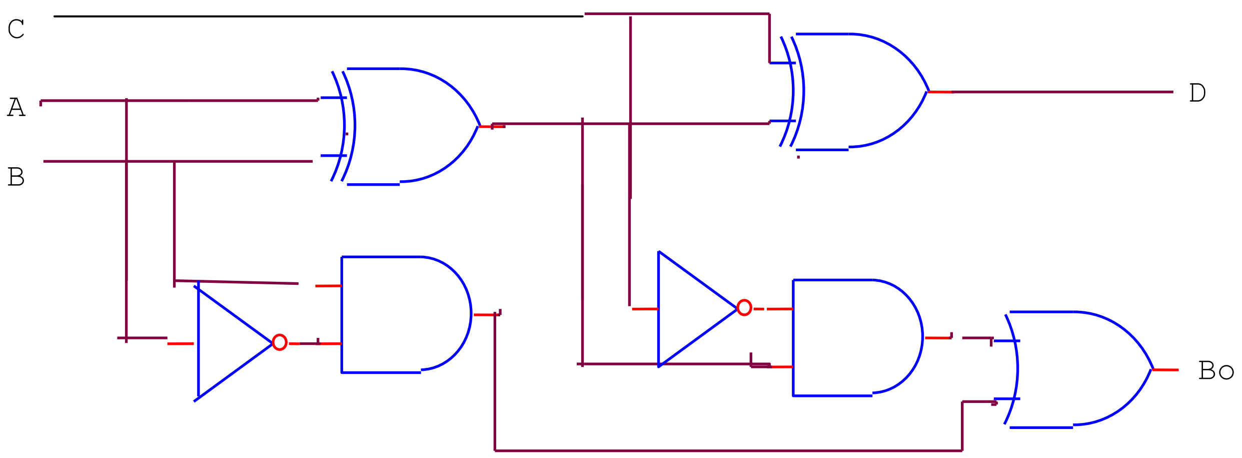
Truth table of Full Subtractor

A	B	C	D	B_o
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



HALF SUBTRACT OR

FULL SUBTRACTOR USING TWO HALF SUBTRACTORS



Result: A half Subtractor was designed and its truth table was verified. A full Subtractor was also designed using half adders and its truth table is verified.

PRECAUTIONS:

1. All the IC's should be handled carefully.
2. All the connection should be tight.
3. Supply should be given after all connections are made.
4. Use IC plucker when remove the IC from Bread Board.

Experiment No. 6

Aim: To design and implement BCD to gray code converter using basic digital logic. Reconvert gray code to BCD and verify the results.

Apparatus required: IC No. 7486, connecting wires and trainers kit.

Theory:

BCD is binary decimal code, the most widely used code. It is 4 bit weighted code having numbers from 0000 to 1001 which represents number from 0 to 9. This is also called 8421 code because weights in 4 bit group reading from left to right are 8421. In this code 1001 is the largest number, i.e. 4 bit group. Others group like 1010, 1011, 1100, 1101, 1110 and 1111 are forbidden. Gray code is unweighted code, used for representation of numbers. It is unweighted because the bits in different positions do not represent positional weights. Each number differs from preceding number by a single bit. It is also known as reflected code.

If $B_3B_2B_1B_0$ and $G_3G_2G_1G_0$ represents 4 bit group in BCD and gray code, then for BCD to gray code:

$G_3 = B_3$, $G_2 = B_2$ XOR B_3, $G_1 = B_2$ XOR B_1 and $G_0 = B_1$ XOR B_0

And for gray to BCD conversion:

$B_3 = G_3$, $B_2 = G_2$ XOR G_3, $B_1 = G_1$ XOR G_2 XOR G_3 and $B_0 = G_1$ XOR G_2 XOR G_3 XOR G_0.

Procedure:

1. Connect the IC on trainer kit and connect the gates in required manner say for BCD to gray code and provide connections for supply and ground.

2. Connect the input and output terminals to required LEDs and switch on the supply.

3. Provide BCD inputs and observe the corresponding gray code outputs.

4. Repeat the same steps for gray code to binary conversion, note down the observations and match both the observations for verification.

Observations:

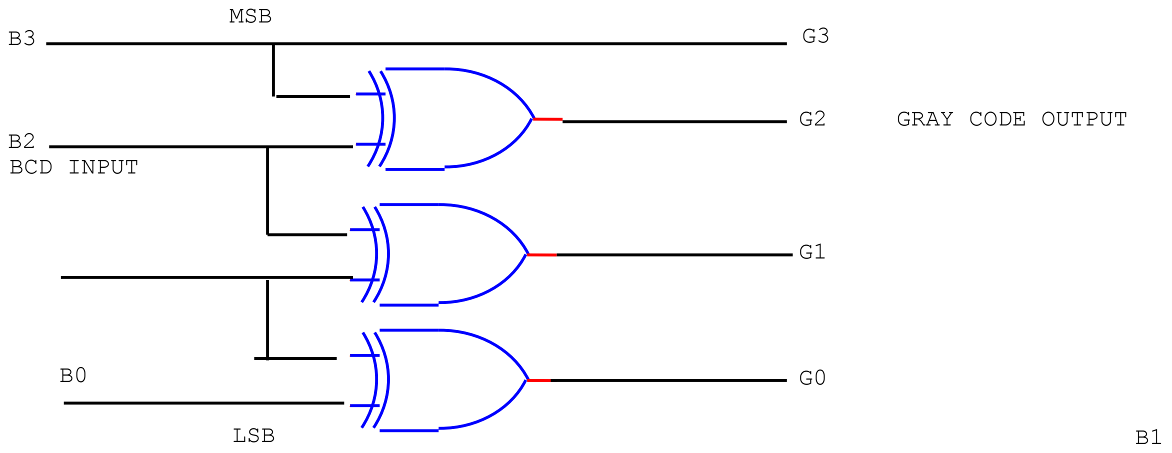
BCD TO GRAY CODE CONVERTER

Decimal Digit	BCD Code Input B_3 B_2 B_1 B_0	Gray Code Output G_3 G_2 G_1 G_0
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1

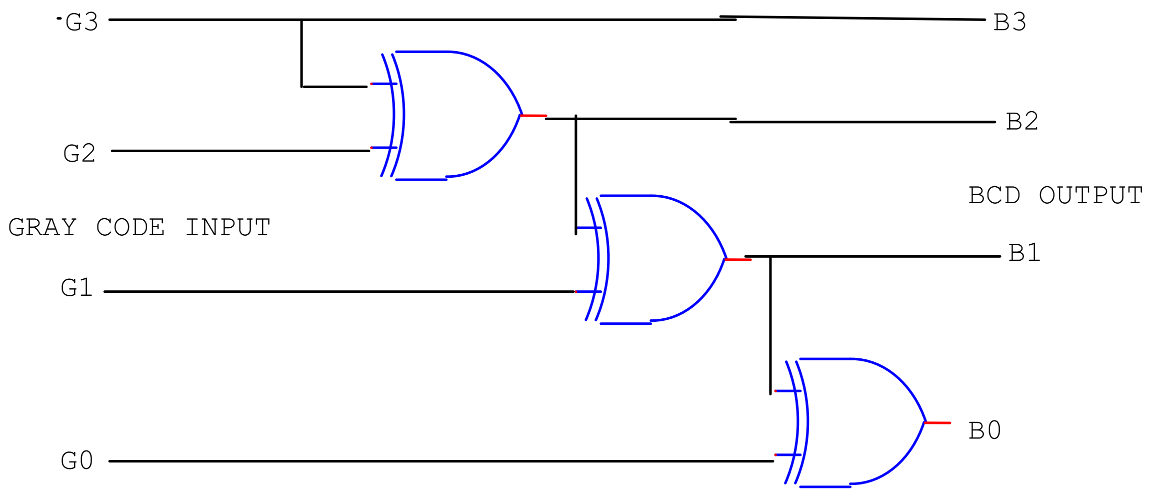
GRAY TO BCD CODE CONVERTER

Decimal Digit	Gray Code Input G_3 G_2 G_1 G_0	BCD Code Output B_3 B_2 B_1 B_0
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 1	0 0 1 0
3	0 0 1 0	0 0 1 1
4	0 1 1 0	0 1 0 0
5	0 1 1 1	0 1 0 1
6	0 1 0 1	0 1 1 0
7	0 1 0 0	0 1 1 1
8	1 1 0 0	1 0 0 0
9	1 1 0 1	1 0 0 1

BCD TO GRAY CODE CONVERTER



GRAY TO BCD CODE CONVERTER



<u>Result:</u> Binary coded decimal input was converted to gray code output and then again reconverted to BCD again and their truth tables were matched for verification.

<u>PRECAUTIONS:</u>

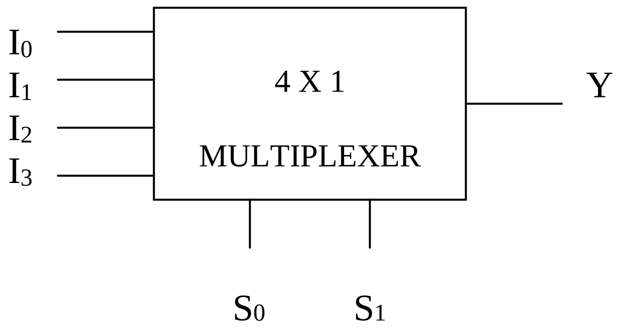
1. All the IC's should be handled carefully.
2. All the connection should be tight.
3. Supply should be given after all connections are made.
4. Use IC plucker when remove the IC from Bread Board.

Experiment No. 7

Aim: To design and implement a 4 X 1 Multiplexer.

Apparatus required: Trainers kit with required integrated circuit and connecting wires.

Theory:



A multiplexer is a special combinational circuit that is one of the most widely used standard circuit in digital design. The multiplexer is a logic circuit that gates one out of several inputs to a single output. The input selected by a set of select inputs. Generally, for 2^n input lines there is one output and n selection lines. Depending upon the digital code applied at the selected input, on out of n data sources is selected and transmitted to a single output channel.

In a 4 X 1 multiplexer, there are 4 input lines (I_0, I_1, I_2 and I_3), 2 selection lines (S_0 and S_1) and one output line (Y). Only one of the inputs is transferred to output depending upon the selection line value. It is implemented using AND gates and OR gate.

Procedure:

1. Connect the gates on the trainer kit in the required manner.

2. Make suitable connections for ground and supply. Connect input and output terminals to required LEDs.

3. Switch ON the supply. For various combinations of input and selection lines, give the input.

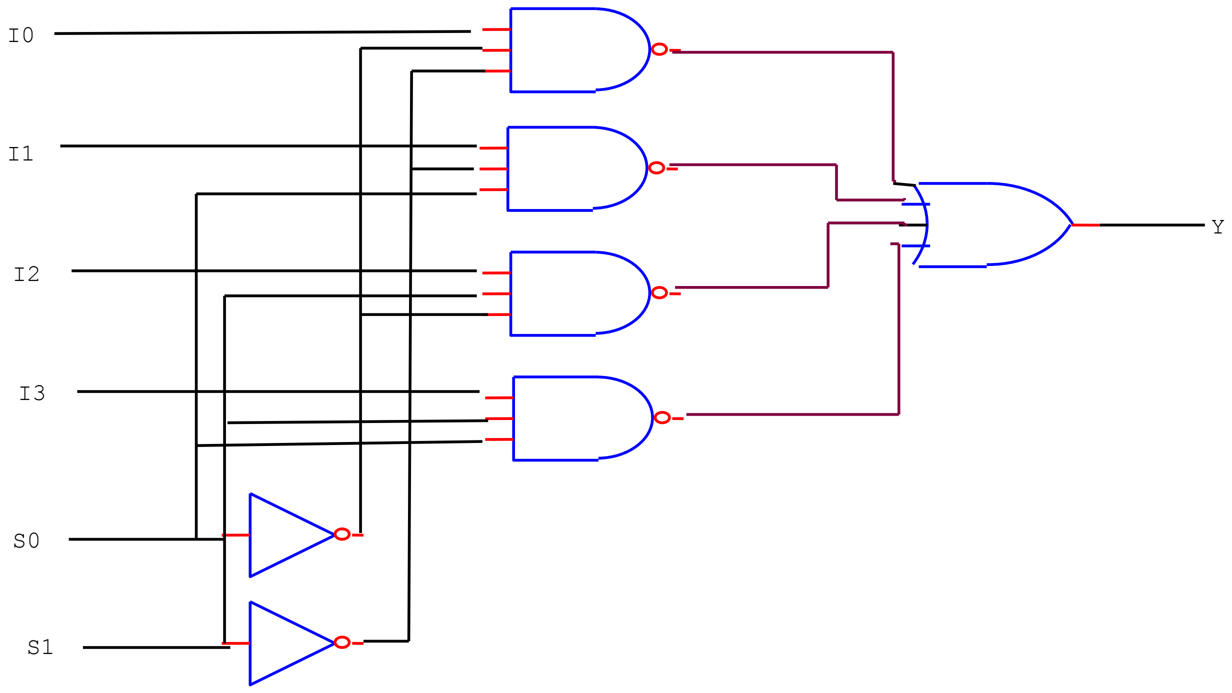
4. Note down the values of output line for various combinations.

Observations:

TRUTH TABLE

S_0	S_1	I_0	I_1	I_2	I_3
0	0	1	0	0	0
1	0	0	1	0	0
0	1	0	0	1	0
1	1	0	0	0	1

4 X 1 MULTIPLEXER



Result: A 4 X 1 multiplexer was designed and implemented using basic digital logic gates and its truth table was noted and verified.

PRECAUTIONS:

1. All the IC's should be handled carefully.
2. All the connection should be tight.
3. Supply should be given after all connections are made.
4. Use IC plucker when remove the IC from Bread Board.

Experiment No. 8

Aim: To design and implement a 3: 8 Decoder

Apparatus required: Trainers kit with required integrated circuit and connecting wires.

Theory:

A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines. If the n bit coded information has unused combination the decoder may have fewer than 2^n output.

An example of decoder is a 3: 8 decoder. The three inputs are decoded into8 outputs each representing one of the minterms of the three input variables. The three inverters provide the complement of the inputs and each one of the eight AND gates generate one of the minterms. A particular application of this decoder is binary to octal conversion. The input variables represent a binary number and outputs represent the eight digits in octal number system. For each possible input combination, there are seven outputs that are equal to 0 and only one that is equal to 1. The output whose value is equal to 1 represents the minterms equivalent of the binary number presently available in input lines.

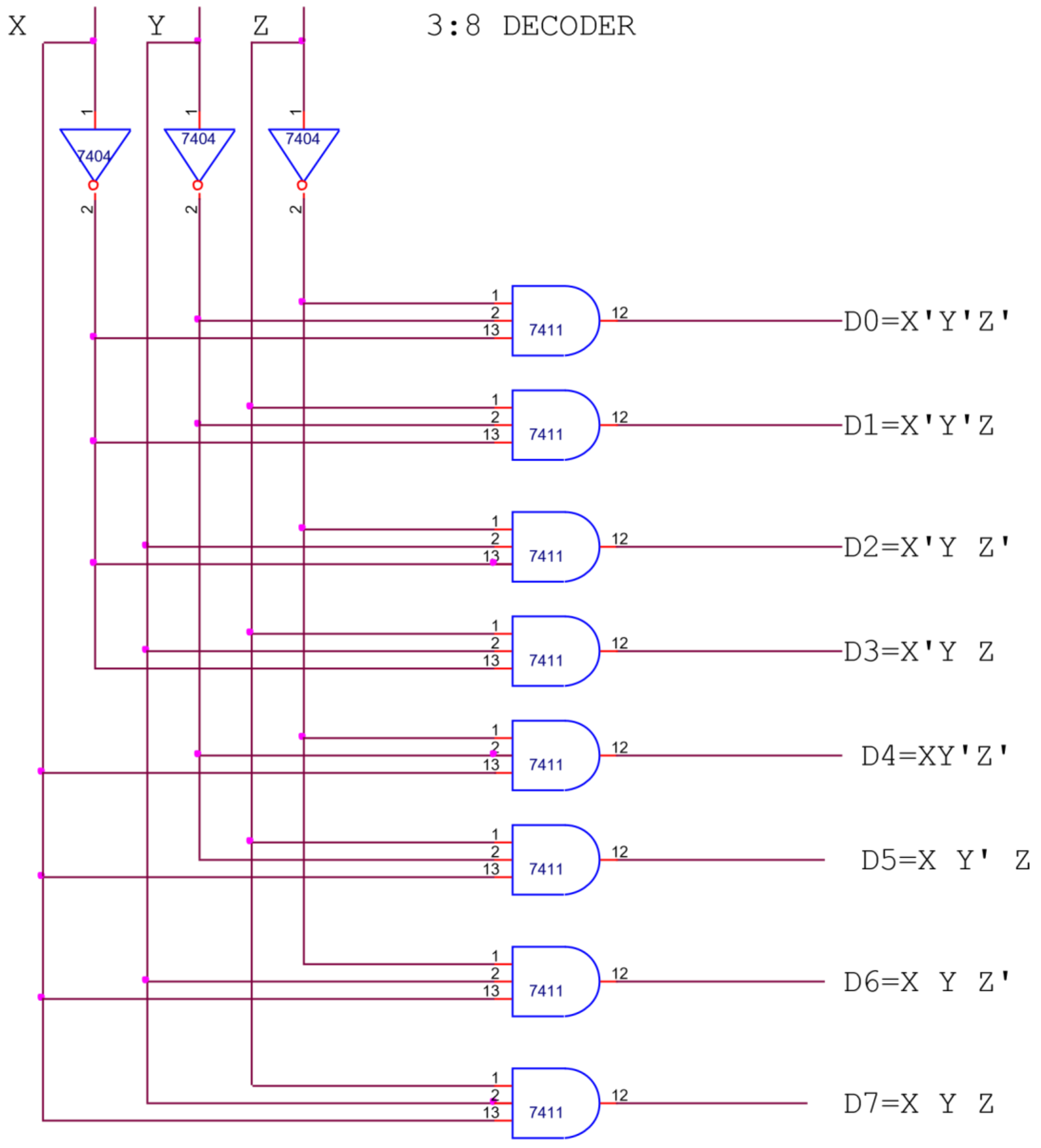
Procedure:

1. Connect the gates on the trainer kit in the required manner.
2. Make suitable connections for ground and supply. Connect input and output terminals to required LEDs.
3. Switch ON the supply. For various combinations of input and selection lines, give the input.
4. Note down the values of output line for various combinations.

Observations:

TRUTH TABLE

INPUTS X Y Z	OUTPUTS							
	D0	D1	D2	D3	D4	D5	D6	D7
0 0 0	1	0	0	0	0	0	0	0
0 0 1	0	1	0	0	0	0	0	0
0 1 0	0	0	1	0	0	0	0	0
0 1 1	0	0	0	1	0	0	0	0
1 0 0	0	0	0	0	1	0	0	0
1 0 1	0	0	0	0	0	1	0	0
1 1 0	0	0	0	0	0	0	1	0
1 1 1	0	0	0	0	0	0	0	1



Result: A 3: 8 Decoder was designed and implemented using basics digital logic gates and its truth table was noted and verified.

PRECAUTIONS:

1. All the IC's should be handled carefully.
2. All the connection should be tight.
3. Supply should be given after all connections are made.
4. Use IC plucker when remove the IC from Bread Board.

Experiment No. 9

Aim: To design and implement BCD to Excess - 3 code converter using basic digital logic. Reconvert Excess - 3 code to BCD and verify the results.

Apparatus required: IC No. 7486, 7408 and 7432, connecting wires and trainers kit.

Theory:

BCD is binary decimal code, the most widely used code. It is 4 bit weighted code having numbers from 0000 to 1001 which represents number from 0 to 9. This is also called 8421 code because weights in 4 bit group reading from left to right are 8421. In this code 1001 is the largest number, i.e. 4 bit group. Others group like 1010, 1011, 1100, 1101, 1110 and 1111 are forbidden.

Excess – 3 code is an unweighted code which is obtained by adding 0011 (3) to each BCD coded value. For example, decimal 2 is coded as 0010 + 0011 = 0101.

For converting ABCD (BCD coded value) to WXYZ then

$W = A + BC + BD$, $X = B'C + B'D + BC'D'$,
$Y = CD + C'D'$, $Z = D'$

and for converting WXYZ BACK TO ABCD, we have

$A = W(X + YZ)$, $B = Y'X' + Y(WZ + WX)$
$C = YZ' + ZY'$, $D = Z'$

Procedure:

1. Connect the IC on trainer kit and connect the gates in required manner say for BCD to Excess - 3 code and provide connections for supply and ground.

2. Connect the input and output terminals to required LEDs and switch on the supply.

3. Provide BCD inputs and observe the corresponding Excess - 3 coded outputs.

4. Repeat the same steps for Excess - 3 code to BCD conversion, note down the observations and match both the observations for verification.

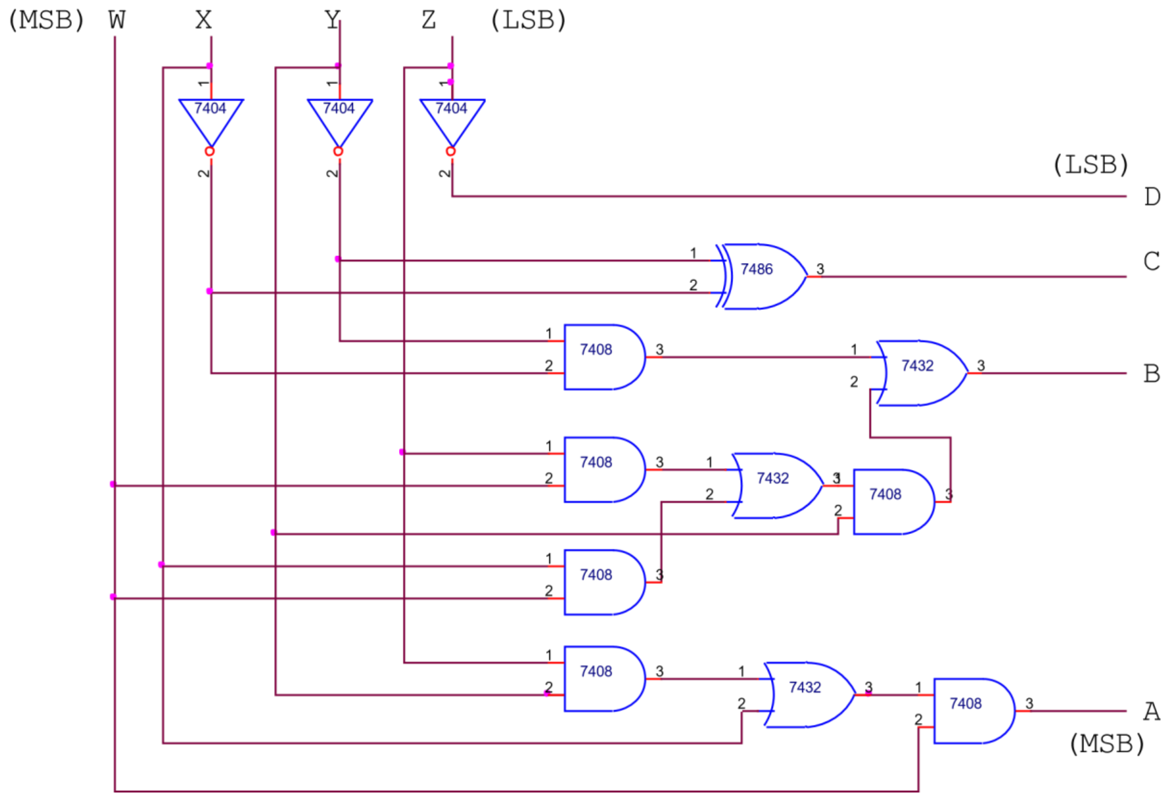
Observations:

BCD TO EXCESS – 3 CODE CONVERTER

Decimal Digit	BCD Input A B C D	EXCESS - 3 Output W X Y Z
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 1
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

EXCESS – 3 TO BCD CODE CONVERTER

Decimal Digit	EXCESS - 3 Input W X Y Z	BCD Output A B C D
0	0 0 1 1	0 0 0 0
1	0 1 0 1	0 0 0 1
2	0 1 0 1	0 0 1 0
3	0 1 1 0	0 0 1 1
4	0 1 1 1	0 1 0 0
5	1 0 0 0	0 1 0 1
6	1 0 0 1	0 1 1 0
7	1 0 1 0	0 1 1 1
8	1 0 1 1	1 0 0 0
9	1 1 0 0	1 0 0 1



EXCESS-3 TO BCD CODE CONVERTER

PRECAUTIONS:

1. All the IC's should be handled carefully.
2. All the connection should be tight.
3. Supply should be given after all connections are made.
4. Use IC plucker when remove the IC from Bread Board.

Experiment No. 10

Aim: To design and implement SR, JK, JKM flip flops.

Apparatus required: Trainers kit with required integrated circuit and connecting wires.

Theory:

A basic digital memory circuit is known as flip flop. It has two stable states as the 1 or set state and 0 or reset state. It can be obtained using NAND or NOR gates. There are various types of flip flops. Some of them are:

1. SR flip flop: This type of flip flop consists of S, R inputs along with clocked pulse input, also the present value of output depends on previous output values. When the clock pulse is 1, then the circuit responds to inputs S and R else the gates G_3 and G_1 inhibited. The combination of S = R = 1 is not allowed as it is indeterminate because here both Q and Q' attain same value.

2. JK flip flop: This is another kind of flip flop. The data inputs are J and K which are AND ed with Q and Q' to obtain S and R inputs (if try to implement it as SR flip flop). In the gates G_3 and G_1, J and K inputs are AND ed with clock pulse and previous outputs. Truth table for JK flip flop is prepared for all possible combinations of JK and for each combination both states of previous outputs have been considered.

3. The Master slave JK flip flop: A Master Slave JK flip flop is a cascade of two SR flip flops with feedback from outputs of the second to inputs of first. Positive clock pulses are applied to first flip flop and are inverted to be applied to second flip flop. When CK = 1, the first flip flop is enabled and second is inhibited and second is enabled. Hence the outputs of second flip flop follow the output of first one. Hence second one is referred as slave and first one as master. Hence this configuration is called Master slave flip flop.

Procedure:

1. For the required flip flop say SR flip flop connect the gates in the required manner.

2. Make the input and output connections in required manner and make connections for supply and ground.

3. Switch on the supply. For the various combinations of input variables more the value of outputs.

4. Repeat the above steps for various flip flops and note the observations for various combinations of input variables.

Observations:

TRUTH TABLE FOR **SR FLIP FLOP**

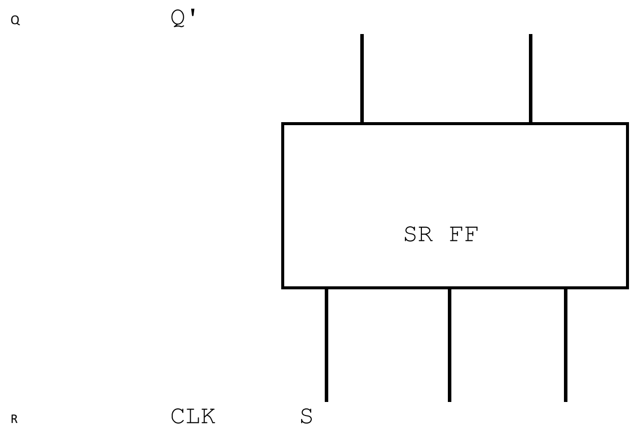
Q_t	S	R	Q_{t+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	*

1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	*

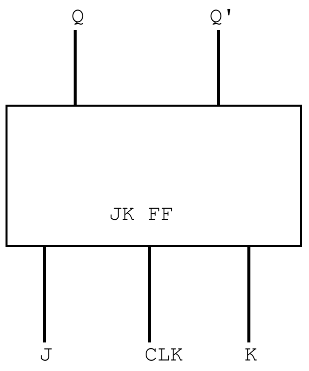
TRUTH TABLE FOR **JK FLIP FLOP**

Q_t	J	K	Q_{t+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

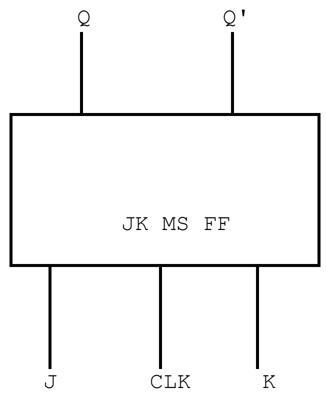
SR FLIP FLOP

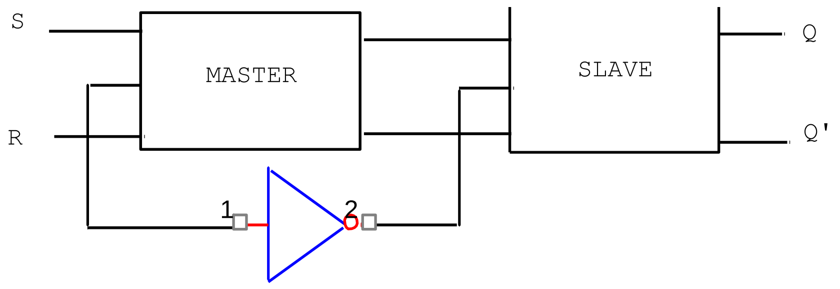


JK FLIP FLOP



MASTER SLAVE JK FLIP FLOP





Result: All the flip – flops - SR, JK, JKMS flip flops were designed and implemented and their characteristic tables were noted.

PRECAUTIONS:

1. All the IC's should be handled carefully.
2. All the connection should be tight.
3. Supply should be given after all connections are made.
4. Use IC plucker when remove the IC from Bread Board.

Experiment No. 11

Aim: To design and implement T and D flip flops.

Apparatus required: Trainers kit with required integrated circuit and connecting wires.

Theory:

1. D flip flop: In this type of flip flop, there is only one input referred to as D input or data input. The output Q_{th} is equal to input D_t when clock pulse is there. This is equivalent to saying that the input data appears at the output end of clock pulse. Thus the transfer of data from the input to the output is delayed and hence named as delay (D) flip flop. The D type flip flop is either used as a delay device or a latch to store 1 bit of binary information.

2. T flip flop: In this type of flip flop, there is only one input referred to as T input. When T = 0, there is no change in the value of output from previous value but when T = 1, the output value toggles from 0 to 1 or 1 to 0. During this period it acts as a toggle switch.

Procedure:

1 . For the required flip flop say D flip flop connect the gates.

2. Make the input and output connections in required manner and make connections for supply and ground.

3. Switch on the supply. For the various combinations of input variables more the value of outputs.

4. Repeat the above steps for T flip flop and note the observations for various combinations of input variables.

<u>Observations:</u>

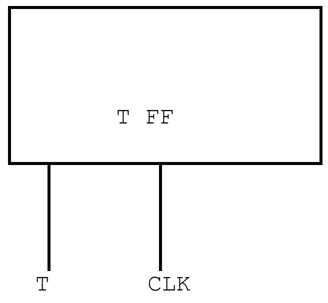
TRUTH TABLE FOR **D FLIP FLOP**

Q_t	D	Q_{t+1}
0	0	0
0	1	1
1	0	0
1	1	1

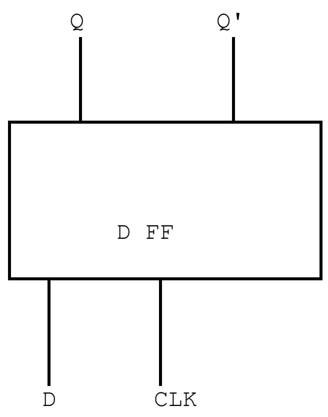
TRUTH TABLE FOR **T FLIP FLOP**

Q_t	T	Q_{t+1}
0	0	0
0	1	1
1	0	1
1	1	0

<u>T FLIP FLOP</u>



D FLIP FLOP



Result: Flip flops T and D were designed and implemented and their characteristic tables were noted.

PRECAUTIONS:

1. All the IC’s should be handled carefully.
2. All the connection should be tight.
3. Supply should be given after all connections are made.
4. Use IC plucker when remove the IC from Bread Board.

Experiment No. 12

Aim: To perform BCD to 7 segment decoding operation.

Apparatus required: Trainers kit, BCD to 7 segment decoder and Connecting wires.

Theory:

A digital display that consists of seven LED segments is commonly used to display decimal numerals in digital systems, e.g. electronic calculators and watches. For using seven segment display device to displaying numerals 0 through 9. For this data has to convert from some binary code to the required for the display. Usually code used is BCD.

In the BCD to Seven segment decoder circuit ABCD are natural BCD codes for numerals 0 through 9. The outputs are a, b, c, d, e, f, g. for representing each numeral certain outputs are to be high. The expressions for all outputs are:

a = B'D' + BD + CD + A b =
B' + C'D' + CD c = B + C' +
D = (B'CD')' d = B'D' + CD'
+ B'C + BC'D e = B'D' + CD'
f = A + C'D' + BC' + BD'
g = A + BC' + B'C + CD'

Procedure:

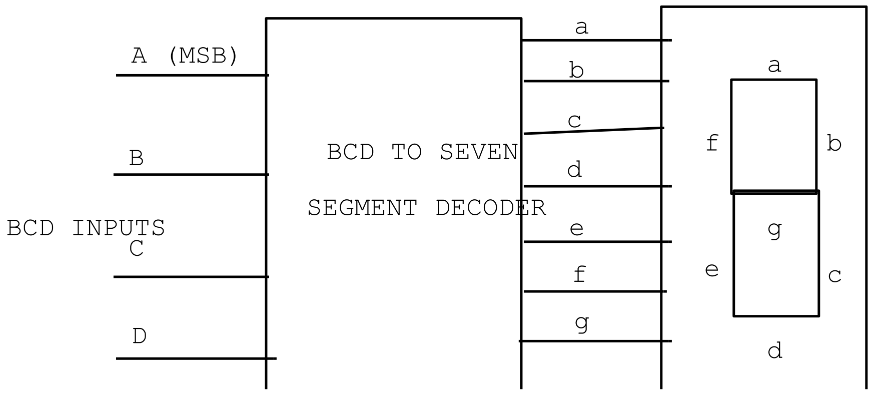
1. Make connections on the BCD to 7 segment decoding trainers kit.
2. Switch ON the supply and now set the inputs to required BCD code value.
3. Observe the display output for each combination of BCD inputs and verify the results.

Observations:

TRUTH TABLE

DECIMAL DIGIT	INPUTS	OUTPUTS
	A B C D	a b c d e f g
0	0 0 0 0	1 1 1 1 1 1 0
1	0 0 0 1	0 1 1 0 0 0 0
2	0 0 1 0	1 1 0 1 1 0 1
3	0 0 1 1	1 1 1 1 0 0 1
4	0 1 0 0	0 1 1 0 0 1 1
5	0 1 0 1	1 0 1 1 0 1 1
6	0 1 1 0	0 0 1 1 1 1 1
7	0 1 1 1	1 1 1 0 0 0 0
8	1 0 0 0	1 1 1 1 1 1 1
9	1 0 0 1	1 1 1 0 0 1 1

BCD TO SEVEN SEGMENT DECODER



Results: BCD to 7 segment decoding operation was performed using BCD to 7 segment decoder kit.

PRECAUTIONS:
1. All the IC's should be handled carefully.
2. All the connection should be tight.
3. Supply should be given after all connections are made.
4. Use IC plucker when remove the IC from Bread Board.